INTELLIGENT INTRUSION DETECTION SYSTEM

Nitin Bhatia
B.E., B.R. Ambedkar University, India, 2000

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SUMMER
2009

INTELLIGENT INTRUSION DETECTION SYSTEM

A Project

by

Nitin Bhatia

Signature removed

____________________, Committee Chair
Dr. Du Zhang

Signature removed

____________________, Second Reader
Prof. Richard Smith

8/12/2009
Date

Student: Nitin Bhatia

I certify that this student has met the requirements for format contained in the University format manual, and that this Project is suitable for shelving in the Library and credit is to be awarded for the Project.

Signature removed

________________________________, Graduate Coordinator 8/17/09

Dr. Cui Zhang Date

Department of Computer Science

Abstract

of

INTELLIGENT INTRUSION DETECTION SYSTEM

by

Nitin Bhatia

Intelligent Intrusion Detection System (IIDS) is a multi-tier enterprise-level Bayesian algorithm based application written in Perl scripting language. A knowledge-based discovery process uses weights added to Snort rules which Bayesian algorithm processes in this IIDS model.

In this project, all alerts identified by Snort through sniffing of network traffic are stored in MySQL database; and a front-end component written in CGI Perl programming language presents these alerts as “True” or “False” options to system administrator. After verification by looking at alerts, system administrator can decide whether a particular attack is “True” or “False” by clicking on one of the buttons. This selection helps Bayesian algorithm learn about various options from stored data in the database such as the attackers IP addresses, types of attack(s) done by attacker(s), number of times attack(s) occurred etc. And from next time on, such attacks automatically get identified as either “True” or “False” as per initial selection; hence making this project unique from other such IDS models based on Bayes algorithm.

The goal of this project is to have an IIDS system which will prevent false positive alerts in the near future, and will also assist identify true attacks and the attacks under progress, as an early detection tool. IIDS does not guarantee any protection from an attack(s) for which there are no rules, also called zero day attack(s).

Signature removed, Committee Chair

Dr. Du Zhang

8/12/2009

Date

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank those who helped me with this project. The great success of this project can be attributed to the guidance and support of many of my friends and colleagues.

I would like to sincerely thank my sponsor, Dr. Du Zhang, for his advice and constant encouragement. I sincerely appreciate the valuable time and effort Dr. Zhang dedicated towards this project.

I would like to also thank Prof. Richard Smith, my second reader, for reading my project report and making beneficial suggestions on content.

TABLE OF CONTENTS

Page

Acknowledgements v

List of Tables ix

List of Figures x

Chapter

1 INTRODUCTION 1

1.1. Background 1

1.2. Purpose 3

1.3. Scope 4

1.4. Related Work 5

1.5. Overview 8

2 TOOLS AND TECHNOLOGIES USED 9

2.1. Snort 9

2.2. Bayesian Algorithm 23

2.3. Linux 26

2.4. Perl 26

2.5. Advantages of Perl with CGI 27

2.6. More Features of Perl 28

2.7. HTML 29

2.8. Bash Script 30

2.9. MySQL 31

2.10. Apache 31

3 DESIGN PRINCIPLES AND IMPLEMENTATION 33

3.1. Overall Architecture 33

3.2. Entity Relationship Diagram 35

3.3. Design Principles 37

3.3.1. Procedural Abstraction 37

3.3.2. Program Structure 38

3.3.3. Software Architecture: Development in Perl 39

4 PERFORMANCE ANALYSIS AND COMPARISON 48

4.1. Comparison with others Work in this Field 48

4.1.1. Active Platform Security through Intrusion Detection Using Naïve Bayesian Network for Anomaly Detection 48

4.1.2. Network-Based Anomaly Intrusion Detection Improvement by Bayesian Network and Indirect Relation 49

4.1.3. Bayesian Event Classification for Intrusion Detection 51

4.1.4. Managing Alerts in a Multi-Intrusion Detection Environment 52

4.2. My Contribution with this Work to IDS Field 53

4.3. Description of Output 54

5 CONCLUSION 64

5.1. Current Drawbacks Associated with the Project 65

5.2. Future Work or Modifications 66

Appendix A – Installation Manual 68

LIST OF TABLES

Page

1. Table 2.1 Snort Default Classifications 19

2. Table 4.1 Comparison of Number of Alerts in IDS and IIDS 63

LIST OF FIGURES

Page

1. Figure 2.1 Components of Snort 22
2. Figure 2.2 Dynamic Website Applications Architecture 30
3. Figure 3.1 Network Setup for Running IIDS 34
4. Figure 3.2 Entity Relationship Diagram for IIDS 35
5. Figure 3.3 Architecture for Running IIDS 37
6. Figure 3.4 Overall Project Architecture 39
7. Figure 3.5 Makefile 40
8. Figure 3.6 Code Snippet Miniserver.pl 41
9. Figure 3.7 Code Snippet of Bayesian Algorithm 42
10. Figure 3.8 Code Snippet of Alerts.pl 44
11. Figure 3.9 Code Snippet of Headers.pl 45
12. Figure 3.10 Code Snippet of Index.pl 46
13. Figure 4.1 Bayesian Working Mechanism on Application Server 55
14. Figure 4.2 Login and Activity Graph of Attackers IP Addresses 56
15. Figure 4.3 Dispersion using Standard Deviation of Alerts 57
16. Figure 4.4 Showing Threat Levels along with Snort Signature 59
17. Figure 4.5 Showing Various Threat Levels from different Attackers IP 60
18. Figure 4.6 Showing Description of Snort Signature with Attackers IP........ 61

19. Figure 4.7 Showing Performance Comparison Between Snort IDS and IIDS alerts in period of Seven Days 62

Chapter 1

INTRODUCTION

1.1. Background

Early-90s witnessed one of the biggest phenomenon witnessed by any industry, better known as Internet era, and various businesses made big success stories using Internet. Right from the days of DRDO ARPA-Network inception in early 1960's, US government and universities gained huge success using Internet. And right from late 1980's, attacks on business and universities started happening over Internet because of non-secured Internet, and business models that lacked even basic security. Various damaging attacks such as SQL slammer worm attack in the year 2002 created havoc over the Internet, and attacks over Internet started becoming prevalent and more damaging in terms of monetary expenses to businesses and universities. Information technology (IT) has been constantly changing, but attacks over Internet have revolutionized the IT industry and various experimental security business models came into existence – each with their own flaws and loopholes. Intrusion Detection System was one of such a revolutionized IT technology that proved to be an efficient and successful security business model. It alerted businesses about attacks in place in real-time, and also came close in providing prior notification for a full-blown attack. However, a drawback with IDS is that they create lots of false alerts, better known as False positives in security terms. At present, simple IDS principle has been implemented in a variety of ways based on various models, such as Decision-tree model [9] used in Symantec Intrusion Prevention System (IPS), and some IDS models are based on anomaly-based techniques. Anomaly detection systems have the advantage that they are able to identify previously unknown attacks [16]. By defining an expected, normal state, any abnormal behavior can be detected, whether it is part of the threat model or not. This capability should make anomaly-based systems a preferred choice [16]. However, the advantage of being able to detect previously unknown attacks is usually over-shadowed by the large number of False positives. This can make the system unusable by flooding and eventually desensitizing the system administrator with large numbers of incorrect alerts. Anomaly-based systems cannot distinguish between anomalous behavior caused by unusual but legitimate actions and activity that is the manifestation of an attack; this leads to the situation where any deviation from normal behavior is reported as suspicious, ignoring potential additional information that might suggest otherwise. Such additional information can be external to the system, received from system health monitors (e.g., CPU utilization, memory usage, process status) or other intrusion detection sensors. Consider the example of IDS that monitors a web server by analyzing the system calls that the server process invokes, a sudden jump in CPU utilization and a continuous increase of the memory allocated by the server process can corroborate the belief that a certain system call contains traces of a denial-of-service attack. Additional information can also be directly related to the models such as the confidence in a model output. Depending on the site-specific structure of input events, certain features might not be suitable to distinguish between legitimate and malicious activity, in such a case, the confidence in the output of the model based on these features should be reduced [3]. Instead of calculating the sum of individual model outputs and comparing the result to a threshold as in anomaly-based system, we utilize a Bayesian decision process [9] to classify input events in IDS. This process allows us to seamlessly incorporate available additional information into the detection decision and to aggregate different model outputs in a more meaningful way. We call this system as Intelligent Intrusion detection system (IIDS), which uses Bayes algorithm to identify False positives and reduce the number of generated alerts.

1.2. Purpose

The main goals of this project are briefed as follows:

- To understand technologies that are used by the industry to implement systems like Intrusion Detection System and come up with a new running model of customized IDS called Intelligent Intrusion Detection System (IIDS). The new method will be designed to cut down on number of False positives (that currently get generated by the best known IDS in market at present and most widely implemented IDS in IT industry called Snort). This would require implementing a two-tier Enterprise system using technologies like Perl, CGI and Bash scripting languages and HTML on Linux based operating system. Linux will also be running Apache web server alongwith various open-source products which are integrated to work together. Please refer to Appendix A for installation and configuration of these modules.

- To implement the Distributed Systems' concept in a real-life IIDS application. That would essentially mean using database tables at run-time, which will be used for both input and output of data from IIDS to web server-based application graphical user interface (GUI). Backend design and implementation of IIDS would be transparent; and information displayed on GUI would change periodically with updated information coming from database.
- Finally, to make this IIDS a complete package at par with any currently available Commercial Off the Shelf (COTS) IPS and IDS software, IIDS can be added into any business model which requires monitoring, early detection and intrusion prevention after modifying some network and internal configuration files as per business model requirements.

1.3. Scope

Intelligent Intrusion Detection System parameters include various open source software which takes inputs from the Snort based IDS engine, and after passing through few customized open source applications and filtering the data, contents are displayed on Apache based web server. The final display on the web server is based on the analysis of who could be potential attacker (in the form of their IP addresses and various other parameters), which attack rule(s) was triggered, rule priorities etc. IIDS also provides the number of attempts and various types of attack(s) done on a target business by a potential attacker. The other customized applications, such as host-based inline-firewall, can be informed to block the potential attackers. Implementation scope of in-line firewall and integration with commercial firewall is beyond this project. This project report can be used as an end-to-end guide for installation, configuration and future development on design principles and implementation of IIDS. This documentation is designed to take a system from bare metal to functional IIDS. IIDS can be customized further, if needed, with strong understanding of working internals of Snort, Linux and scripting in Perl and Bash.

1.4. Related Work

There is not much work that can be found to be done on IDS especially using Bayesian decision process. Bayesian approach such as na¨ıve Bayes and Bayesian networks based on decision process which can be followed for making a system smart and self sustainable by learning. Out of the four closest references that I found with my work, one of them talks about Bayesian event classification for IDS [3], while the other talks about Bayesian model for real-time IDS [12]. The first reference, Bayesian event classification, deals with distributed denial of service (DDOS) attack by system calls on Linux and on Sun Solaris server. This model has compared decision trees and na¨ıve Bayes with Bayesian networks [3]. We follow another good comparison done by "Nahla ben Amor, Salem Benferhat, and Zied Elouedi" on "Na¨ıve Bayes vs. Decision Trees in IDS" done in the year 2004 [9]. In their model they talk about the classification capability of a na¨ıve Bayesian network, which is identical to a threshold-based system that computes the sum of the outputs obtained from the child nodes [9]. This is due to the fact that all models (i.e., the child nodes) operate independently and only influence the probability of the root node. This single probability value at the root node can be represented by a threshold in traditional approaches. In addition, the restriction of having a single parent node complicates the incorporation of additional information. This is because variables that represent such information cannot be linked directly to the nodes representing the model outputs [9]. Following their model approach, Bayesian networks based on decision process is my approach too to implement IIDS. Another work done by "Kruegel, Mutz, Robertson and Valeur" in "Bayesian event classification for Intrusion detection" [3] fall short in showing implementation in real time networks.

To study Bayesian networks' real-time implementation I followed work done by "Ricardo S. Puttini, Zakia Marrakchi, and Ludovic Mé" in their implementation of "Real-time IDS model following Bayesian classification" [12]. In their approach, they present a new design of an anomaly IDS. Design and development of the IDS are considered in three main stages: normal behavior construction, anomaly detection and model update. A parametrical mixture model is used for behavior modeling from reference data. The associated Bayesian classification leads to the detection algorithm. A continuous model parameter re-estimation is discussed as a possible heuristic for model update [12]. The focus in their model is to make it self-learning and improve on alerting, which is a shared goal for IIDS project too. They have not, however, considered rapidly changing rules and their model consists of set of fixed rules. Compared to their model my implementation has daily rules' updates which requires user intervention for correcting False positives. There is another good study done by "Byung Rae Cha and Dong Seob Lee" on "Network based anomaly IDS using Bayesian networks and shows indirect relation with anomaly and alert" done in the year 2007 [2]. Their model focuses on IDS alerts coming from attack on FTP service running on a system. My work is very similar to their model, with few differences in terms of graphical user interface etc. Both our models have successful implementation in real-time along with integration of open source IDS Snort. One major difference in my approach, the so-called parameter re-estimation is done by assigning or continuously modifying Snort rule priority manually to cut down on False positives and with each re-estimation, MySQL database would be purged, hence forcing Bayesian decision based network to start calculating again, making it a knowledge-based IDS. This method ensures detection of the occurrence state or action sequences that had been previously identified to be an intrusion, hence making it a practical IIDS implementation unlike Network-Based Anomaly IDS.

From the year 2006 onwards, Bayesian decision tree methods' implementation and research have slowly moved away from IDS systems to actual data prevention systems; and based on this new trend, new business models have come into existence. One such model based on Bayesian decision tree is implemented in various agent based "Data Security Suites" COTS software; wherein a centralized server will talk to end point agents installed on various clients' Operating systems (OS) to finger-print various pattern or signature-matching files on those Operating systems, and then generate a report for system administrator. Various policies can be enforced through the agents running as end-point clients on systems such as 'do not copy', 'modify' or 'save' such pattern matching files, etc. This model has led to protection of copy of confidential files or documents, stealing credit card and social security numbers (SSN) from end–users' systems. This is still an evolving field and we still have a long way to achieve complete data security or data loss prevention.

1.5. Overview

Chapter 2 will discuss tools and technologies used to design and implement the Intelligent Intrusion Detection System. Chapter 3 will include the design principles and implementation details of Intelligent Intrusion Detection System. Chapter 4 will include the performance analysis and comparison. Finally, chapter 5 will be the conclusion followed by the Appendix and References.

Chapter 2

TOOLS AND TECHNOLOGIES USED

This project mainly used open source software technologies and was developed in two-tier model. Front-end was developed to do dynamic HTML-based CGI-Perl program running on Apache web server. For the second-tier, bash scripting was used along with the various Perl modules. One such Perl module, the DBI connector, is for reading and writing to MySQL database running on an enterprise Red Hat Linux platform used in this project. Snort IDS, which was part of second-tier, sniffed network traffic and analysed it using its' rules for various alerts and attacks. The key concept to make this project successful was Bayesian Algorithm which was written in Perl and implemented with various Perl modules to create Intelligent Intrusion Detection System

2.1. Snort

Snort Intrusion Detection System is the main components in IIDS architecture. Snort is an open source network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly-based inspection methods. Snort is a very widely deployed intrusion detection and prevention technology worldwide and has become the de-facto standard of the industry.

Snort can be configured with other software, both commercial off the shelf (COTS) and open source software, under various deployable features such as inline-Firewall (to drop network packets), Honey Pot, and a sniffer [10] etc. In this project, Snort's feature of sniffer is used along with some modifications in configuration files to make Snort work as an Intrusion Detection System. Snort has its own IDS signatures and rule sets which can be easily modified and customized as per the requirements of the business. This aspect has also been used in this project with set manually assigned priority as shown in Table 2.1.

Snort sniffs network packets from a network card running in a promiscuous mode, the packets are then passed on to Snort's engine which decodes the packets and matches them against configured and customized rule-sets. Once the first rule is matched against the decoded packet, a corresponding alert is raised and stored in the database. In this project, Snort is configured to use MySQL database and stores all alerts in the appropriate table under MySQL database.

Here is an example of a Snort rule followed with brief description:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET4000 (msg:"EXPLOIT Alt-N Security
Gateway username buffer overflow attempt"; flow:established, to_server;
content:"username=";nocase;isdataat:450,relative;content:!"&";within:450;content:!"|0A|
"; within:450; metadata:policy balancedips drop, policy connectivity-ips drop, policy
securityips drop; reference:url,secunia.com/advisories/30497/; priority:1;
classtype:attempted-admin; sid:13916; rev:2;)
```

Rule Header consists of rule actions and rule options.

- Rule Actions:

The rule header contains the information that defines the who, where, and what of a packet, as well as what to do in the event that a packet with all the attributes indicated in the rule should show up. The first item in a rule is the rule action. The rule action tells

Snort what to do when it finds a packet that matches the rule criteria. There are five available default actions in Snort: alert, log, pass, activate and dynamic. In addition, if you are running Snort in inline mode, you have additional options which include drop, reject, and sdrop.

- Alert – generates an alert using the selected alert method, and then logs the packet.
- Log – log the packet.
- Pass – ignore the packet.
- Activate – alert and then turn on another dynamic rule.
- Dynamic – remain idle until activated by an activate rule, then act as a log rule
- Drop – make iptables drop the packet and log the packet.
- Reject – make iptables drop the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol sis UDP.
- Sdrop – make iptables drop the packet but do not log it.

You can also define your own rule-types and associate one or more output plugins with them. You can then use the rule-types as actions in Snort rules.

This example will create a type that will log to just tcpdump.

```
ruletype suspicious
{
Type log
output log_tcpdump: suspicious.log
}
```

This example will create a rule type that will log to syslog and a MySQL database:

```
ruletype redalert
{
type alert
output aler_syslog; LOG_AUTH LOG_ALERT
output database: log, mysql, user=snort dbname=snort host=localhost
}
```

- Protocols:

The next field in a rule is the protocol. There are four protocols that Snort currently analyzes for suspicious behavior – TCP, UDP, ICMP, and IP. In the future, there may be more such as ARP, IGRP, GRE, OSPF, RIP, IPX, etc

- IP Addresses:

The next portion of the rule header deals with the IP addresses and port information for a given rule. The keyword 'any' may be used to define any address. Snort does not have a mechanism to provide host name lookup for the IP address fields in the rules file. The addresses are formed by a straight numeric IP address and a CIDR [10] block. The CIDR block indicates the netmask that should be applied to the rule's address and any incoming packets that are tested against the rule. A CIDR block mask of /24 indicates a Class C network, /16 a Class B network, and /32 indicates a specific machine address. For example, the address/CIDR combination 192.168.1.0/24 would signify the block of addresses from 192.168.1.1 to 192.168.1.255. Any rule that used this designation for, say, the destination address would match on any address in that range. The CIDR designations give us a nice short-hand way to designate large address spaces with just a few characters. There is an operator that can be applied to IP addresses, the negation operator, This operator tells Snort to match any IP address except the one indicated by the listed IP address. The negation operator is indicated with a '!.' For example, an easy modification to the initial example is to make it alert on any traffic that originates outside of the local net with the negation operator as shown below:

alert tcp !192.168.1.0/24 any ->192.168.1.0/24 111 (content: "|00 01 86 a5|"; msg: "external mountd access";)

This rule's IP addresses indicate any tcp packet with a source IP address not originating from the internal network and a destination address on the internal network.

You may also specify lists of IP addresses. An IP list is specified by enclosing a comma separated list of IP addresses and CIDR blocks within square brackets. IP lost may not include spaces between the addresses. An example of an IP list in action is in the shown alert:

tcp ![192.168.1.0/24,10.1.1.0/24] any -> \

[192.168.1.0/24,10.1.1.0/24] 111 (content: "|00 01 86 a5|"; msg: "external mountd access";)

- Port Numbers:

Port numbers may be specified in a number of ways by including 'any' ports, static port definitions, ranges, and by negation; 'any' ports are a wildcard valuebecause they represent literally any port, static ports are indicated by a single port number, such as 111 for portmapper, 23 for telnet, or 80 for http, etc. And, port ranges are indicated with the range operator:. The range operator may be applied in a number of ways to take on different meanings. For example, you wanted to log everything except X Windows ports, then you could do something like the rule below:

log udp any any -> 192.168.1.0/24 1:1024 log udp

traffic coming from any port and destination ports ranging from 1 to 1024.

log tcp any any -> 192.168.1.0/24:6000

log tcp traffic from any port going to ports less than or equal to 6000

log tcp any :1024 -> 192.168.1.0/24 !6000:6010

- The Direction Operator:

The direction operator '->' indicates the orientation or direction of the traffic to which the rule applies. The IP address and port numbers on the left side of the direction operator is considered to be the traffic coming from the source host, and the address and port information on the right side of the operator is the destination host. There is also a bidirectional operator which is indicated with a '<>' symbol. This indicates Snort to consider the address/port pairs in either the source or destination orientation. This is handy for recording/analyzing both sides of a conversation, such as telnet or POP3 sessions. For example, bidirectional operator being used to record both sides of a telnet session is shown below. Also, note that there is a no '<-' operator. In Snort versions before 1.8.7, the direction operator did not have proper error-checking and many people used an invalid token. The reason the '<-' does not exist is so that rules always read consistently.

log tcp !192.168.1.0/24 any <> 192.168.1.0/24 23

- Activate/Dynamic Rules:

Activate/dynamic rule pairs give Snort a powerful capability. You can now have one rule activate another when its action is performed for a set number of packets. This is very useful if you want to set Snort up to perform follow-on recording when a specific rule goes off. Activate rules act just like alert rules except that they have a required option field ":activates". Dynamic rules act just like long rules, but they have a different option field ":activated by". Dynamic rules have a second required field as well: count. Activate-rules are just like alerts but also tell Snort to add a rule when a specific network event occurs. Dynamic rules are just like long rules except that they are dynamically enabled when the activate rule id goes off.

```
activate tcp !$HOME_NET any -> $HOME_NET 143 (flags: PA; \
content: "|E8C0FFFFFF|/bin"; activates: 1;  msg: "IMAP buffer overflow!"; )
dynamic tcp !$HOME_NET any -> $HOME_NET 143 (activated_by: 1; count: 50;)
```

Above activate/dynamic rule example indicates Snort to alert when it detects an IMAP buffer overflow and collect the next 50 packets headed for port 143 coming from outside $HOME NET headed to $HOME NET. If the buffer overflow happened and was successful, there's a very good possibility that useful data will be contained within the next 50 (or whatever) packets going to that same service port on the network, so there is value in collecting those packets for later analysis.

- Rule Options:

Rule options form the heart of Snort's intrusion detection engine combining ease-of-use with power and flexibility. All Snort rule options are separated from each other using the semicolon (;) character. Rule option keywords are separated from their arguments with a colon (:) character. There are four major categories of rule options.

- General: These options provide information about the rule but do not have any
- Payload: These options ook for data inside the packet payload and can be inter-related.
- Non-payload: These options look for non-payload data.
- Post-detection: These options are rule specific triggers than happen after a rule has 'fired."

- Msg:

The 'msg' rule option tells the logging and alerting engine the message to print along with a packet dump or to an alert. It is a simple text string that utilizes the backslash – '\'- as an escape character to indicate a discrete character that might otherwise confuse Snort's rules parser (such as the semi-colon -';' - character). Format msg: "<message text>";

- Reference

The reference keyword allows rules to include references to external attack identification systems. The plug-in currently supports several specific systems as well as unique URLs. This plug-in is to be used by out plugins to provide a link to additional information about the alert produced. More information can be found at http://www.snort.org/pub-bin/sigs-search.cgi/ for a system that is indexing descriptions of alerts based on the sid.

Format : reference: <id system>,<id>; [reference: <id system>,<id>;

Example

```
alert tcp any any -> any 7070 (msg:"IDS411/dos-realaudio"; \
flags:AP; content:"|fff4 fffd 06|"; reference:arachnids,IDS411;)
alert tcp any any -> any 21 (msg:"IDS287/ftp-wuftp260-venglin-linux"; \
flags:AP; content:"|31c031db 31c9b046 cd80 31c031db|"; \
reference:arachnids,IDS287; reference:bugtraq,1387; \
reference:cve,CAN-2000-1574;)
```

- Gid

The gid keyword (generator id) is used to identify what part of Snort generates the event when a particular rule fires. For example, gid 1 is associated with the rules' subsystem and various gids over hundred are designated for specific preprocessors and the decoder in Snort. See '/etc/<Snort directory>/generators' in the source tree for the current generator ids in use.

NOTE: The gid keyword is optional and if it is not specified in a rule, it will default to one, and the rule will be part of the general rule subsystem. To avoid potential conflict with gids defined in Snort (that for some reason aren't noted it etc/generators), it is recommended that a value greater than 1,000,000 be used. For general rule writing, it is not recommended that the gid keyword be used. This option should be used with the sid keyword. The file etc/gen-msg.map contains more information on preprocessor and decoder gids.

Format: gid: <generator id>;

Example: This example is a rule with a generator id of 1000001.

```
alert tcp any any -> any 80 (content:"BOB"; gid:1000001; sid:1; rev:1;)
```

- Sid

The sid keyword is used to uniquely identify Snort rules. This information allows output plugins to identify rules easily. This option should be used with the rev keyword.

• <100 Reserved for future use

• 100-1,000,000 Rules included with the Snort distribution

• >1,000,000 Used for local rules

The file sid-msg.map contains a mapping of alert messages to Snort rule IDs. This information is useful when post-processing alert to map an ID to an alert message.

Format: sid: <snort rules id>;

Example

This example is a rule with the Snort Rule ID of 1000983.

```
alert tcp any any -> any 80 (content:"BOB"; sid:1000983; rev:1;)
```

- Rev

The rev keyword is used to uniquely identify revisions of Snort rules. Revisions, along with Snort rule id's, allow signatures and descriptions to be refined and replaced with updated information. This option should be used with the sid keyword.

Format: rev: <revision integer>;

Example

This example is a rule with the Snort Rule Revision of 1.

```
alert tcp any any -> any 80 (content:"BOB"; sid:1000983; rev:1;)
```

- Classtype

The classtype keyword is used to categorize a rule as detecting an attack that is part of a more general type of attack class. Snort provides a default set of attack classes that are used by the default set of rules it provides. Defining classifications for rules provides a way to better organize the event data Snort produces.

Format: classtype: <class name>;

Attack classifications defined by Snort reside in the classification.config file. The file uses the following syntax:

config classification: <class name>,<class description>,<default priority>

These attack classifications are currently ordered with three default priorities: A priority of 1 (high) is the most severe and 3 (low) is the least severe.

Table 2.1 below shows various classifications as per their priorities.

Classtype	Description	Priority
attempted-admin	Attempted Administrator Privilege Gain	high
attempted-user	Attempted User Privilege Gain	high
kickass-porn	SCORE! Get the lotion!	high
policy-violation	Potential Corporate Privacy Violation	high
shellcode-detect	Executable code was detected	high
successful-admin	Successful Administrator Privilege Gain	high
successful-user	Successful User Privilege Gain	high
trojan-activity	A Network Trojan was detected	high

Table 2.1 Snort Default Classifications

unsuccessful-user	Unsuccessful User Privilege Gain	high
web-application-attack	Web Application Attack	high
attempted-dos	Attempted Denial of Service	medium
attempted-recon	Attempted Information Leak	medium
bad-unknown	Potentially Bad Traffic	medium
default-login-attempt	Attempt to login by a default username and password	medium
denial-of-service	Detection of a Denial of Service Attack	medium
misc-attack	Misc Attack	medium
non-standard-protocol	Detection of a non-standard protocol or event	medium
rpc-portmap-decode	Decode of an RPC Query	medium
successful-dos	Denial of Service	medium
successful-recon-largescale	Large Scale Information Leak	medium
successful-recon-limited	Information Leak	medium
suspicious-filename-detect	A suspicious filename was detected	medium
suspicious-login	An attempted login using a suspicious username was detected	medium
system-call-detect	A system call was detected	medium
unusual-client-port-connection	A client was using an unusual port	medium

Table 2.1 : Snort default Classifications continued ...

web-application-activity Access to a potentially vulnerable web application medium

icmp-event Generic ICMP event low

misc-activity Misc activity low

network-scan Detection of a Network Scan low

Table 2.1 : Snort default Classifications continued ...

Example

tcp-connection A TCP connection was detected very low

```
alert tcp any any -> any 80 (msg:"EXPLOIT ntpdx overflow"; \
dsize: >128; classtype:attempted-admin; priority:10 );
alert tcp any any -> any 25 (msg:"SMTP expn root"; flags:A+; \
content:"expn root"; nocase; classtype:attempted-recon;)
```

NOTE: The classtype option can only use classifications that have been defined in snort.conf by using the config classification option. Snort provides a default set of classifications in classification.config that are used by the rules Snort provides.

- Priority

The priority tag assigns a severity level to rules. A classtype rule assigns a default priority defined by the config classification option) that may be overridden with a priority rule.

Format: priority: <priority integer>;

Example

```
alert TCP any any -> any 80 (msg: "WEB-MISC php attempt"; flags:A+; \
content: "/cgi-bin/php"; priority:10;)
```

In general, these options provide information about the rule, but do not have any affect during detection of payload. These options look for data inside the packet payload and can be inter-related to non-payload; they look for non-payload data post-detection. The options are rule-specific triggers that happen after a rule has fired. The msg rule option tells the logging and alerting engine the message to print alongwith a packet dump or an alert. It is a simple text string that utilizes the '\' as an escape character to indicate a discrete character that might otherwise confuse Snort's rules parser (such as the semi-colon ';' character).

Format : msg: "<message text>";

The complete process of Snort decode is show in Figure 2.1. Snort can be configured to run on distributed clients or servers, and it can sniff packets on various networks if configured in distributed environment.

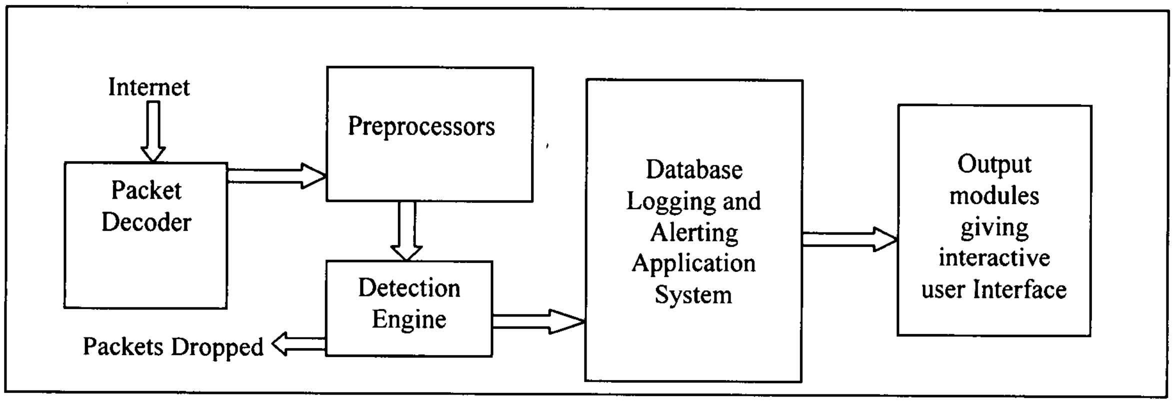

Internet
Preprocessors
Packet Decoder
Database Logging and Alerting Application System
Output modules giving interactive user Interface
Detection Engine
Packets Dropped

Figure 2.1 Components of Snort

The flexibility of Snort running in distributed environment alongwith MySQL database allows them to be configured in a fashion which will log all alerts coming from various Snort engines to a common MySQL database.

NOTE: This project has a scope to run multiple Snort engines in a distributed environment which can be configured to talk to one common MySQL database with slight modification in configuration file of Snort and Linux. Installation manual under Appendix A shows how connection from Snort using “Snort.conf” file can be made with MySQL database from multiple Snort engines or sensors.

In this project, Snort is running as a server-side component which needs to run in a highly available, fault-tolerant, transactional, and secure form multi-user environment to maintain the integrity of alerts logged in the MySQL database. The application server provides this high-end server-side environment after configuration of various open source software to parse and filter False positive alerts, and it provides runtime updates from the MySQL database to Perl-CGI based front-end to manage and monitor various attacks on an enterprise.

2.2. Bayesian Algorithm

Bay was a Mathematician in early 19th Century. His theorem, better known as Bayes theorem states, “if you have an event and you want to know the likely occurrence of that event happening, you can determine a probability of that event by looking at a subset of randomly selected parts of those groups." A Bayesian network is used to model a domain containing uncertainty [6, 8]. It is a directed acyclic graph (DAG) where each node represents a discrete random variable of interest. Each node contains the states of the random variable that it represents a conditional probability table (CPT). The CPT of a node contains probabilities of the node being in a specific state given the states of its parents. The parent-child relationship between nodes in a Bayesian network indicates the direction of causality between the corresponding variables. That is, the variable represented by the child node is causally dependent on the ones represented by its parents [3].

For example, you have an election and you have a randomly selected group of voters who are going to vote on a proposition in either Yes or No. You pick up a randomly selected sub-group (community) and determine what is the likely occurrence of the Yes vote. Then you can use Bayes Algorithm to calculate the probability of Yes vote for the largest group (State). Suppose, if in a smaller community of 10 people there were 7 Yes votes and 3 No votes. Then from Bayes theorem, there is a probability of 89% for a Yes vote overall. This is how future prediction in cases of voting and elections are made by various News agencies and Networks.

Now let's take an example of Bayes theorem usage in the field Computer Science. A very good and well know example would be of Spam-filters, they have various names. Before taking up Bayes theorem as an alternative to cut down on number of False positives in IDS, I studied Bayes algorithm usage in Spam-filtering and also looked into the code of an Open Source Software known as "Spam assassin". This is how a Spam-Filter works:

To determine a message is a Spam, Bayes filter takes a subset, that is inside message it will look at the individual word and assign it a probability based on its occurrences, then it will look at a combination of words and assign them a probability of occurring together, it will also look at various other parameters such as the color in the HTML E-Mail, URL's in the E-Mail, also the placement of individual word. And, based on these parameters it will assign themthe probability to determine they are Spam or not. For example, let's look at the successful implementation of Bayes Algorithm in Yahoo E-mail Service. The appearance of word 'Viagra' in E-mail is assigned some probability. The combined probability of words 'Viagra' and 'URL' appearing in the same E-mail gets even a higher probability. But, a message is marked as a Spam E-mail when an E-mail contains words 'Viagra', 'URL' and 'Buy' as the combined probability of these words passes the set global probability which identifies it as a Spam E-mail. Bayes theorem when applied to Spam with its probability theory flags out real junk mail and is less likely to create False positives.

In this project, the probability is assigned manually to all the rules of Snort as shown in Appendix B under 'Class Bayes' and 'Signature Bayes.' The groups under Snortcontain similar type of rules classified on the basis of various attacks rules' classifications or categories. There is a global probability that is manually set initially to 0.5 to flag the attacks at different threat levels after calculating attacker IP address, number of occurrences, etc. using Bayes theorem. The administrator can select whether a flagged message is a False positive or a True attack. This process results in training the Bayes filter in IIDS just like Spam E-mail messages. Administrator can also modify global probability and mark more alerts as False positives to further cut down on generated results. But, a word of caution to anybody who wants to do that as it might result in flagging true attacks as False positives.

2.3. Linux

Red Hat Enterprise Linux is the platform of choice for this project. Linux provides flexibility to easily configure and customize various open source software. Further, Linux provides us an ability which none of the current available Operating System provides, such as to control the processes, drivers, kernel customization, setting up of customized security model. etc. Programming and scripting can be done in almost any language on Linux as it comes with default installation of almost all commonly known programming languages. Apache web server has been chosen for use in this project as it can be easily configured and customized on Linux platform.

Some of the key implementations of IIDS project on Linux is that it easily provides a controlled front-end communication with the second-tier software after using some scripting and programming language. Linux also supports key features of this project such as:

- **Dynamic data frequency updates integration.** For example, alert data from MySQL database builds the front-end web server page dynamically, or returns a previously built page if it is still up to date with the help of Perl-CGI and Apache modules available on Linux.

2.4. Perl

Perl is a scripting language that lets you mix regular and static HTML with dynamically generated HTML. Many web pages that are built by CGI programs are mostly staticwith the dynamic part limited to a few small locations. But most CGI variations, including Perl make you generate the entire page via your program, even though most of it is always the same.

2.5. Advantages of Perl with CGI

- **Vs. Pure Servlets.** Servlets are more robust and dynamic, but it is more convenient to write and to modify regular HTML with Perl than to have thousands of HTML statements.
- **Vs. Server-Side Includes (SSI).** SSI is a widely supported technology for including externally defined pieces into a static web page, Perl is better because it lets you use connections among various programs instead of a separate program to generate that dynamic part. Besides, SSI is only intended for simple inclusions, and not for real programs that use form data, make database connections and the like.
- **Vs. JavaScript.** JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client's environment. With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can't access server-side resources like databases, catalogs, pricing information, and the like.
- **Vs. Static HTML.** Regular HTML cannot contain dynamic information. Perl is easy and convenient that it is quite feasible to augment HTML pages that only

benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances.

2.6. More Features of Perl

- **Efficient.** With traditional CGI, a new process is started for each HTTP request. If the program does a relatively fast operation, the overhead of starting the process can dominate the execution time. With Perl, processes stay up, and a lightweight Perl process also called thread handles each request. Perl threads do not rely on heavyweight operating system processes to handle requests. Similarly, in traditional CGI, if there are *N* simultaneous requests to the same CGI program, then the code for the CGI program is loaded into memory N times. With Perl, there is a single process which helps in monitoring, debugging, making optimizations such as caching previous computations, keeping database connections open which eventually interacts with a CGI page for generating output in this project.
- **Convenient.** Perl has an extensive infrastructure called classes for automatically parsing and decoding HTML form data, reading and setting HTTP headers, handling cookies, tracking sessions, and many other such utilities.
- **Powerful.** Perl lets you easily do several things that are difficult or impossible with regular CGI. For example, Perl processes can talk directly to the web server (regular CGI programs can't). This simplifies operations that need to look up

images and other data stored in standard places. Perl can also share data among other threads and make database connection pools easy to implement. Perl can make connection with network devices, and Perl with CGI can also maintain information from request to request simplifying things like session tracking and caching of previous computations.

- **Portable.** Programs written in Perl can run virtually unchanged on Apache, Microsoft IIS, or Web-Star [12].
- **Inexpensive.** There are a number of free or relatively inexpensive web servers available that are good for personal use or for low-volume websites. However, with the exception of Apache, most commercial-quality web servers are relatively expensive. Adding Perl support to the web server is, however, generally free or cheap.

2.7. HTML

A website usually includes a collection of Hypertext Markup Language (HTML) pages. HTML is a markup language that controls how text appears on web pages. There are two types of web pages: static and dynamic. Static, as the name implies, refers to the web pages that are saved on a disk and passed to the user's web browser without any changes. A dynamic web page is one that has content that is changed by some program or script before it is displayed to the user. In the early days of the internet, almost all web pages were static, but slowly this trend has changed. The Internet boom of late 90's changed the landscape of the World Wide Web with the introduction of electric-commerce also called as 'e-commerce.' For example, businesses would want too show their customers their inventory online. For this, a static page would not serve very well since inventory ischanging throughout the day. This is where dynamic website comes into play. A dynamic website typically consists of two parts: a scripting language to display the data and a database to store the data. Figure 2.2 below shows the relationship between the two components.

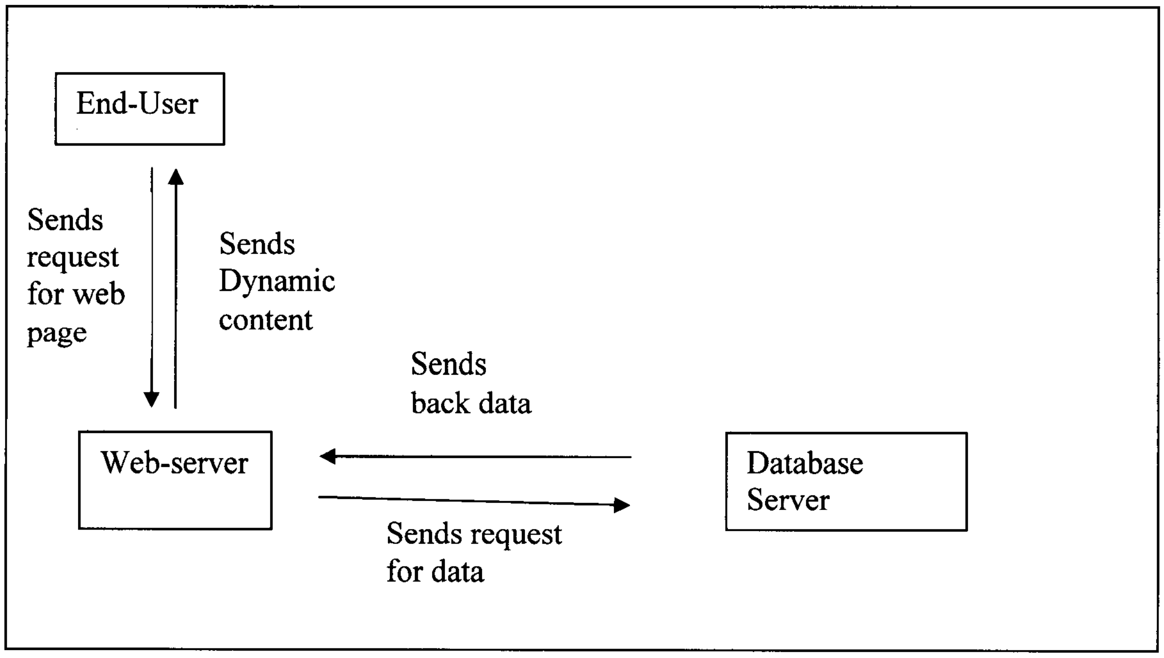


Figure 2.2 Dynamic Website Applications Architecture

2.8. Bash-Script

Bash Scripting is mostly used for the Unix/Linux automation of tasks or processes. Further, bash scripting can be used for validation and integrity of second-tier MySQL database. So in the context of this project, it has limited but significant usage. It makes the web page loading as easy and as quickly possible while keeping the database very lightweight as it purges old alerts from MySQL database.

2.9. MySQL

MySQL is a database management system (DBMS). A database can be thought of as a computerized collection of information and an organized body of related information. A DMBS allows the user to easily insert, delete, update, and query data. MySQL is referred to as a relational database system based on the relational model. Relational model is a data model based on predicate logic and set theory. For example, in IIDS project, MySQL stores the alert-data in multiple tables instead of one big table. These multiple tables are related to each other, hence the term relational. Multiple tables offer speed and efficiency when it comes to accessing the tables. To query the data, we use Structured Query Language (SQL) passed through Perl. There are many advantages to MySQL that make it a very popular choice for dynamic websites.

- Open Source – MySQL is an open source database. Open source also provides great support possibility. It is usually much easier to get help from a web message board than it is to call the company that makes professional software.
- Fast – MySQL offers a bare bone database system that is suitable for a low traffic website. It does not provide many of the advanced features that a commercial DMBS will provide, but it makes up for that with its speed and can be further customized for efficiency.

2.10. Apache

Apache is an open-source web server. In a very competitive application and web server market where software giants like Microsoft, Oracle, Adobe and IBM rule with their overall applications development platform suites and web servers, Apache still manages to thrive because of its open-source origins. It provides a reasonable lightweight to heavy application and web server integration with easy implementation through slight modification in configuration files that come embedded with an Apache implementation. Apache can be configured to run with virtually any database product like Oracle and MySQL and it supports almost all programming languages and scripting languages.

Chapter 3

DESIGN PRINCIPLES AND IMPLEMENATION

The Design of this project consists of the following major tasks:

- Overall architecture; in terms of design and implementation of Firewall, placement of the system, and monitoring of IIDS system where Snort sniffs packets.
- Entity relationship diagrams for Snort and Bayesian tables in MySQL database.
- Description and implementation of IIDS Project and relationship between various Bayesian components written in Perl.

3.1. Overall architecture

IIDS is placed behind Firewall sniffing inbound and outbound traffic leaving and coming into the corporate network. This is the best placement scenario for an IIDS as Firewall will block unwanted traffic from coming into the network which Firewall administrator has deemed unwanted already. Hence, IIDS will only be seeing and monitoring the anomalies coming from the traffic which is permitted in and out of the network.

Figure 3.1 gives a detailed setup for setting up IIDS in network. In this setup Snort, MySQL, Apache web server is running on a Linux server. There is a network connection at layer 2 only which means traffic coming out of Firewall and going to Firewall is

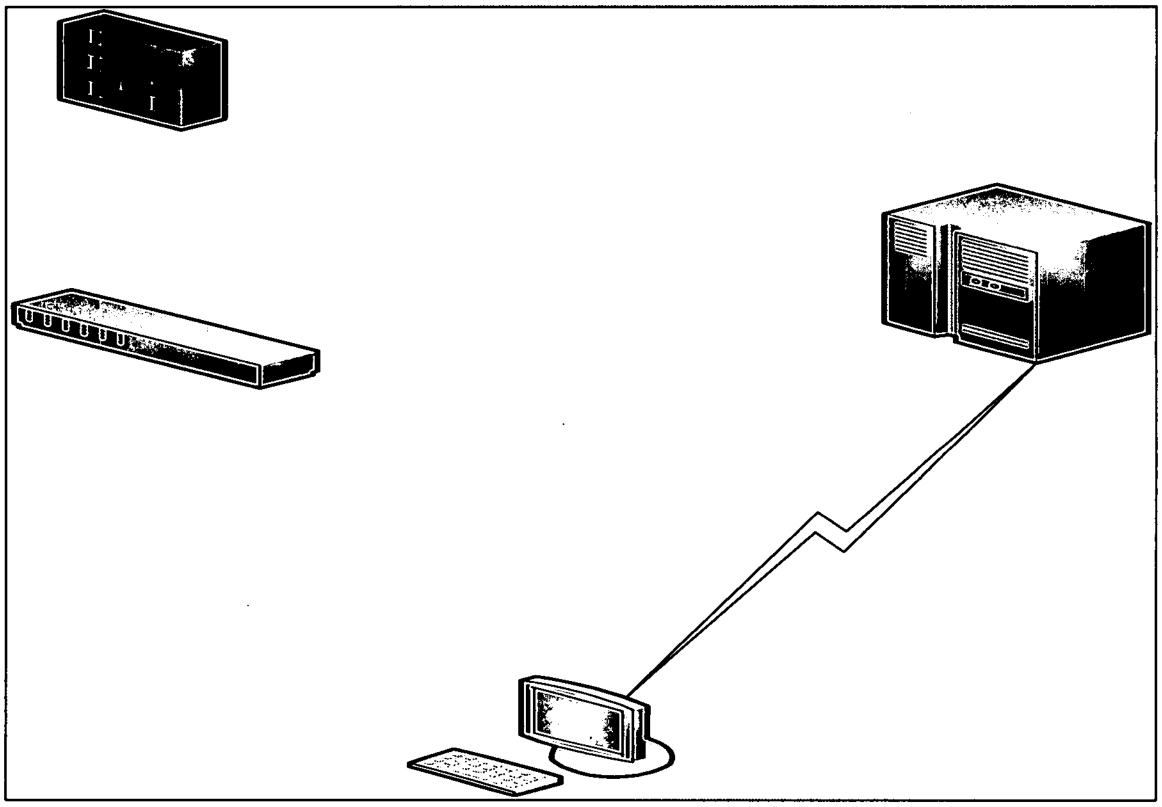


Figure 3.1 Network Setup for Running IIDS

by a switch running in a monitoring state on one of its interface which it connected to a promiscuous interface on IIDS server. Basically, IIDS interface running in promiscuous mode receives all traffic which is passing through the Firewall. IIDS is running with two interfaces one is running in promiscuous mode which is used for sniffing the traffic and other is running normally with an assigned IP. IIDS interface with an IP is used for reporting and for managing purpose. IIDS reports can be viewed from any workstation for management purpose after authentication. Access on IIDS for management and for reporting purpose is controlled by host based Firewall on IIDS called IP-tables. IP-tables permit or deny access based on your workstation IP address when you are trying to connect to IIDS.

3.2. Entity Relationship Diagram

There is only one database management system used in IIDS which is MySQL. Snort logs to various tables in Snort database in MySQL database. There are three new tables created in MySQL under Snort database for storing various Bayesian parameters. The following schema shows E.R. between various tables in MySQL database.

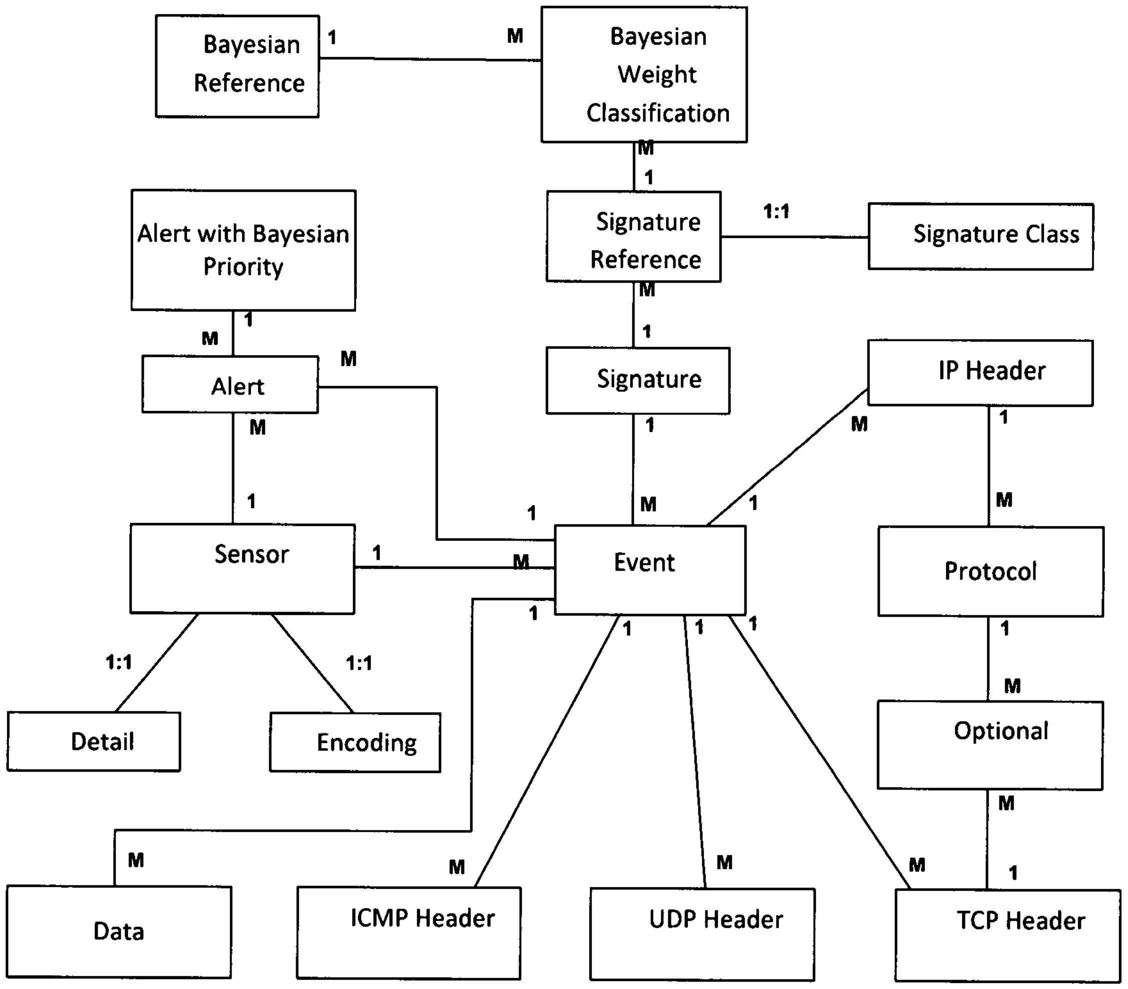


Figure 3.2 Entity Relationship Diagram for IIDS

Entity Relationship schema compromises of above shown entities only. Everything is an entity as data in Snort database under MySQL can be broken down to individual entities.

Looking at relationships among various entities an event is classified as an alert with an appropriate priority. Process for classification is simple as shown in Figure 3.2 E.R. diagram. An event is classified with an associated Snort signature(s), each Snort signature has a corresponding Snort reference based on the severity, each Snort reference has a Bayesian weight value such as 0.5 for general alerts, 0.7 for severe, 0.9 for critical and so on defined by administrator based on the nature of severity. Weight value of 0.5 is a threshold value. An event occurring number of times will have more weight than threshold value and will be classified accordingly. This is where False positive alerts are reduced or cut down as administrator defines the Bayesian weight or conditional probability table (CPT) [3] for an event which will be called an alert. Bayesian reference value is the set of database of events happening with a probability of one, i.e. any event of associated Bayesian weight occurring once. If an event continues to happen with same entities set over a period of time, Bayesian reference value keeps increasing hence, showing this event as high probability or high priority event. An appropriate Snort rule based priority is also defined for each event. If a Bayesian high priority event has an equally high Snort rule priority than that event is classified as critical and requires attention. If a Bayesian high priority event has equally low Snort rule priority, than either Snort rule priority requires modification or event based on Snort rule should be marked as False positive in front-end GUI. Critical alerts with high Bayesian priorities are shown as red in reports and with lesser Bayesian priorities are shown as yellow or blue and so on in reports.

3.3. Design Principles

This section presents design principles behind our IIDS model. As you can see from above E.R. model there is a modularity based approach in each component. We essentially have the following architecture of the application ready and running in IIDS model.

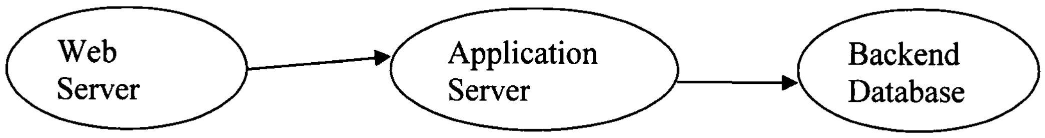


Figure 3.3 Architecture for Running IIDS

Main focus of this project was devoted on to data abstraction after forming an overall procedural abstraction as shown below and software architecture programming was done in Perl following a bottom-up approach.

Each component of this project mainly part of MySQL database has it's modularity or unique identity right in each Snort and in Bayesian table, which leads us into data abstraction of each entity as shown in Figure 3.2. Each entity in E.R. diagram has collection of its own data set and almost all of the Perl scripts written in this IIDS project consist of SQL statements which queries various Snort database tables in MySQL to obtain different types of information to be displayed on client web browser.

3.3.1. Procedural Abstraction

This section presents our IIDS model in main sequence of events also called procedural abstraction. The idea is to build a behavior model that takes into account multiple alerts and allows a posteriori Bayesian classification of data as part of the detection algorithm. A reference audit data set representing the normal system behavior is used to create the model with a learning procedure, i.e. when IIDS system starts up for the first time alerts triggered by Snort rules are presented to system administrator through a front end, for system administrator to decide whether they are False or True positives alerts. Hence, making Bayesian based IIDS system to learn about various attacks as either True or False positives after the initial selection.

Following section shows the detailed design of all the components that went into design of this IIDS. IIDS logic is based on Bayesian algorithm implemented in Perl and most of the Perl modules work in tandem to extract data from MySQL database to display various kinds of information on an IIDS website using CGI and HTML.

3.3.2. Program Structure

For implementing this phase, a Bayesian program in Perl along with other open source Perl modules and packages was used with a bottom-up programming approach. IIDS Bayesian module essentially intercepts calls from the backend MySQL database using above procedural structures in the form of various SQL querries. Snort engine components talk to the backend Snort database directly and populates tables accordingly as per alerts classifications. IIDS intercepts or obtains information in terms of input from information stored in Snort database coming from original Snort tables to determines or learns various types of attack patterns and also by analyzing various Bayesian tables implemented in Snort database. IIDS learns and filters information to the front-end system based on the events stored in the database under various classifications such as assigned probability, priority of the rule(s) which generated the alert(s), number of times an alert triggered from a particular source IP, type of event etc. This integration of these Snort tables along with Bayesian tables and various Perl components in IIDS makes this whole Snort based system smarter. Figure 3.4 shows how this is achieved in the 2-tier architecture of the design process.

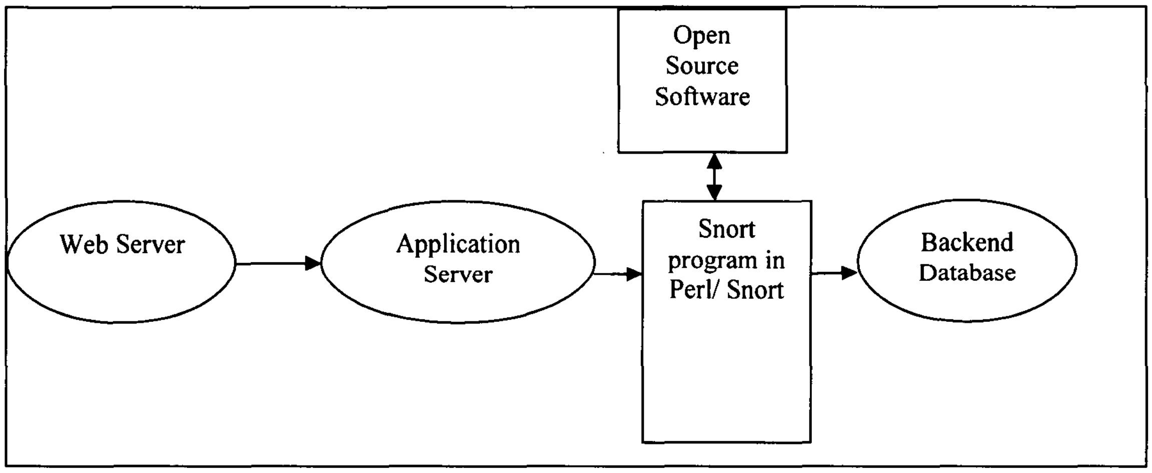


Figure 3.4 Overall Project Architecture

3.3.3. Software Architecture: Development in Perl

Figure 3.5 gives complete description of makefile. This makefile provides detailed description of root directory where Snort is installed with the bin directory path, port for Snort to connect to the system and an IP address by which Snort pages can be accessed on client machine. The key feature in the default installation of Snort is port 666. Make sure port 666 is open on the system on which Snort is begin installed, one can verify if the port is open or not by using "nmap -p" option.

All these installation parameters of Snort can be changed in a Makefile including the port, IP address and installation path. Key generation part is added for authentication using certificates on the system. It can be removed from the Makefile if you do not want to generate a key for the system on which Snort is installed. But, then make sure to remove the SSL module from other *.pl.

NOTE: Complete code can be found under Appendix B; it presents complete structured details.

```
# License: This software is under GPL license, Date: 06 2005
ROOT = /opt/snort
CONFIGPATH = /etc/snort
BIN=snort.sh
BINPATH=/usr/sbin
certname = snort
life = 730
keylength = 1024
listen_port = 666
listen_ip = 127.0.0.1
#KEY_REPOSITORY = $(CONFIGPATH)/ssl_key
SSL_PROGRAM = /usr/bin/openssl
install : dir snort selfsign perlgd conf run by
dir :       mkdir -p $(ROOT)
            mkdir -p $(CONFIGPATH)
snort : @echo "snort install"
            chmod 755 $(ROOT)
#keygen:
#           @if test -f $(KEY_REPOSITORY)/$(certname).key ; then \
                        echo 'key $(KEY_REPOSITORY)/$(certname).key already exist'; \
                         else \ echo 'Generating RSA key $(KEY_REPOSITORY)/$(certname).key'; \
                        $(SSL_PROGRAM) genrsa -out $(KEY_REPOSITORY)/$(certname).key
```

Figure 3.5 Makefile

Figure 3.6 shows miniserv.pl which has most of the Perl modules for web server configuration such as CGI available modules, web server port and web directory modules, user authentication, sockets connections, TCP connections, verifying SSL connections, and setup of Syslog for logging information Perl modules. Latest version of miniserv.pl can be obtained from Webmin.

```
# CGI or normal file
local $rv;
if (&get_type($full) eq "internal/cgi") {
        # A CGI program to execute
        $envtz = $ENV{"TZ"};
        $envuser = $ENV{"USER"};
        $envpath = $ENV{"PATH"};
        $envlang = $ENV{"LANG"};
        foreach (keys %ENV) { delete($ENV{$_}); }
        $ENV{"PATH"} = $envpath if ($envpath);
        $ENV{"TZ"} = $envtz if ($envtz);
        $ENV{"USER"} = $envuser if ($envuser);
        $ENV{"OLD_LANG"} = $envlang if ($envlang);
        $ENV{"HOME"} = $user_homedir;
        $ENV{"SERVER_SOFTWARE"} = $config{"server"};
        $ENV{"SERVER_NAME"} = $host;
        $ENV{"SERVER_ADMIN"} = $config{"email"};
        $ENV{"SERVER_ROOT"} = $config{"root"};
        $ENV{"SERVER_PORT"} = $port;
        $ENV{"REMOTE_HOST"} = $acpthost;
        $ENV{"REMOTE_ADDR"} = $acptip;
        $ENV{"REMOTE_USER"} = $authuser if (defined($authuser));
        $ENV{"BASE_REMOTE_USER"} =$baseauthuser if ($authuser ne $baseauthuser);
```

Figure 3.6 Code Snippet Miniserver.pl

Figure 3.7 shows Snort.pl snippet showing the implementation of Bayesian algorithm in Perl. This file also sets up various attack groups into high, low and medium groups, which are implemented using hash and key feature in Perl. Snort.pl can easily be said heart of IIDS, as most of the parameters passed to web pages using CGI for display to users are generated using this Perl program. Snort.pl also contains various SQL statements, which helps in display of hostility, dispersion graphs etc. One part of this file contains metric database for all 'Snort Rules Classes' grouped together, which can be customized to any value between 0 and 1, with 1 begin the critical alert and 0 begin the low attack. Metric database helps in calculation of the severity of attack(s) by using Bayesian Algorithm based on the individual class metric. It is strongly recommended to update this database in Snort.pl as and when new classes are added in Snort rules as mentioned in Chapter 2, you can certainly call it a flat-file database within Snort.pl.

```
# Get bayes probability
sub get_bayes {
                my($FILE)="signature_bayes";
                my $line;
                my @buffer;

        open(PTR_FILE,$FILE) or die "Problem opening $FILE !";

        while($line=<PTR_FILE>) {
                @buffer = split (/:/,$line);
####################### Bayes equation ###########################
if (($buffer[1])||($buffer[2])) {
                        $bayes{$buffer[0]} = 2*(($buffer[1] * $global_prob) / ( ($buffer[1]*
$global_prob) + ($buffer[2] * (1-$global_prob))));
                }
}       close(PTR_FILE); }
```

Figure 3.7 Code Snippet of Bayesian Algorithm

There is another flat-file database in Snort.pl, which is known as Severity database. This database maps the severity levels of metric database coming from Bayesian metrics which were initially based on Snort rule priority. For example, low severity for a Snort rule in a particular class will map to 0.01 for the same class group rule in metric database and so on. In case of attack(s) when a Snort rule with high priority is trigerred it will map to high severity class group in metric database, in Bayesian based IIDS if this attack is occurring for the first time, IIDS will present this attack on front-end for True or False positive classification to system administrator. In case(s) where Bayesian has seen such attack(s) before coming from same data sets such as same IP, if same rule is triggered etc, then Bayesian based IIDS looks up at the previous selection of either True or False positive done by system administrator and then either alerts or discards that alert as per previous selection. In Perl these values are mapped using Hash feature by passing Key and Value pair to map with each other. Both severity and metric flat-file database should match and synchronize with each other else program will fail due to inconsistent key and value pair values. The global attack probability is set to 0.5 any attack above that is defined as "Aggressive" or "Hostile" in hostility metric threshold.

Figure 3.8 shows Alerts.pl. This file contains various SQL queries to show various types of alerts and to delete unwanted alerts contained in the MySQL database manually. It also contains the priorities of 'Snort Rules Classes' containing different types of rules grouped together. These alerts are displayed using CGI implementation in Perl, visible to user from their web browser. It is required to update this file with any new addition of Snort Class in Snort.pl. Alerts.pl also contains various Perl modules such as DBI module to connect to database, for authenticating user before changes can be made to the MySQL database.

```
# Main program

if ($TRI eq ""){
        $TRI = 3;
}

$HUMAN_DATE=sapiens(time());
&build_report_query();
&head();
&form();

if ($TYPE_DEL eq "6"){
        &delete_all();
        $TYPE_DEL = "1";
}

if ($TARGET_DEL ne ""){

        if ($TYPE_DEL eq "1"){
                &del_by_source();
        }

        if ($TYPE_DEL eq "2"){
                &del_by_dest();
        }

        if ($TYPE_DEL eq "3"){
                &del_by_date();
        }
        if ($TYPE_DEL eq "4"){
                &del_by_signature();
        }
                if ($TYPE_DEL eq "5"){
                &del_by_sensor();
        } }
if ($PASS ne ""){&report();}&foot();
```

Figure 3.8 Code Snippet of Alerts.pl

Figure 3.9 shows Header.pl, which gets attackers IP addresses header information from MySQL database and generates a report for users in CGI, viewable on any web browser. This program obtains various header information from TCP protocol stored in Snort TCP header table in Snort database in MySQL. Following type of information is provided from TCP table: Source port, Destination port, Window size, Various Flags, and Sequence number. Similarly, it provides information on UPD protocol with the following information, Source port, and Destination port. And for ICMP protocol the information is: ICMP Type, ICMP Code, and ICMP ID.

```
# Main program
&head();
$HUMAN_DATE=sapiens(time());
&report();
&foot();
sub report{        my ($i, $DBH, $STH, $NUMBEROW, $NUMBERFIELDS, @COLUMN, $REF,
                          $PROTOCOL, $SERVICE, $ICMPTYPE, $ICMPCODE,$SIGID);
my $ALERT_QUERY="
SELECT event.timestamp,signature.sig_name,signature.sig_sid,sig_class.sig_class_name
```

Figure 3.9 Code snippet of Headers.pl

Figure 3.10 shows Index.pl, which mainly contains the complete information and memory load processes for hostility and dispersion criterias. This information is obtained using various SQL statements running on MySQL database written within index.pl. Most part of this file is similar to Snort.pl and should be updated if Snort.pl is modified as this file contains metric database for all 'Snort Rules Classes' group, which can be customized to any value between 0 and 1, as mention under Snort.pl.

All these modules form critical and core design components of IIDS project. This how all these modules are linked together. Initially makefile links these modules together as it installs and create directories, installs openssl, miniserv Perl module, snort program, and

CGI Perl modules, along with it modifies path and changes permissions for directories so that they are accessible by only root user.

```
use strict;
my %weight; # this is the metric database
# The metric can be changes based  on events
$weight{'icmp-event'} = "0.01";
$weight{'misc-activity'} = "0.02";
$weight{'network-scan'} = "0.03";
$weight{'not-suspicious'} = "0.001";
$weight{'protocol-command-decode'} = "0.02";
$weight{'string-detect'} = "0.01";
$weight{'unknown'} = "0.01";
$weight{'bad-unknown'} = "0.2";
$weight{'attempted-dos'} = "0.02";
$weight{'attempted-recon'} = "0.02";
$weight{'denial-of-service'} = "0.5";
$weight{'misc-attack'} = "0.5";
$weight{'non-standard-protocol'} = "0.1";
$weight{'rpc-portmap-decode'} = "0.1";
$weight{'successful-dos'} = "0.1";
$weight{'successful-recon-largescale'} = "0.1";
$weight{'successful-recon-limited'} = "0.1";
$weight{'suspicious-filename-detect'} = "0.1";
$weight{'suspicious-login'} = "0.1";
$weight{'system-call-detect'} = "0.1";
```

Figure 3.10 Code snippet of Index.pl

Miniserver.pl sets up front end integration with CGI and web server. Figure 3.7 provides detailed implementation of Bayesian algorithm in Perl along with how Bayesian algorithm queries MySQL database and learn whether an attack is True or False positive. Figure 3.7 also shows how Bayesian algorithm stores identified True or False positive alerts in various variables in Perl in the form of array's and finally how it connects to MySQL database using DBI module written in Perl. Figure 3.8 provides detailed description how alerts are received from MySQL database and are provided to front end using miniserv Perl module and Figure 3.9 shows how formatting is done on front end to display information. Once alerts are displayed on front end web page, an administrator can select them as either True or False positive which are then stored in an array in Perl by module in Figure 3.7. This is also how Bayesian algorithm remembers if next time similar attack will be either a True or False positive as identified by administrator or a user.

Chapter 4

PERFORMANCE ANALYSIS AND COMPARISON

This chapter provides information on the results including performance analysis and comparison of improved performance with the current Snort based IDS and Bayesian based IIDS. This chapter also provides detailed screenshots from the output of this project.

4.1. Comparison with others Work in this Field

The work done in this project is challenging, as I have attempted to cut down False positives by successfully implementing and modifying a Snort based IDS. This work is not presented as just another theoretically based project. I have used four different closest matching studies to the evaluate my work done on IIDS project

4.1.1. Active Platform Security through Intrusion Detection Using Naïve Bayesian Network for Anomaly Detection

Research work done by "Abdullah A. Sebyala, Temitope Olukemi and Dr. Lionel Sacks" in their above titled paper presents Bayesian model to detect anomalies on system processes such as CPU resources etc. Their research work focuses on approach which allows third party executable codes (proxylet) to be deployed into the network, which creates a big security risk [1]. In their model they use Bayesian networks to study behavior of an application on a system. If an executable code is doing something different to a system or to an application and if some change is detected which was not expected behavior of the executable, then it is flagged as an anomaly as per Bayesian networks in their model. Their model can be used or can be associated more closely to host based intrusion detection systems, better known as (HIDS).

My working model is an example of network based intrusion detection system which can be also referred as (NIDS). In my model, I use Bayesian algorithm to detect anomalies after initially making Bayesian learn what anomalies are, which is done by system administrators manual selection of True and False positives alerts. Once my model understand what are True or False alerts it cuts down on future False alerts. Hence, my approach in my model is different compared to above study.

4.1.2. Network-Based Anomaly Intrusion Detection improvement by Bayesian Network and Indirect relation

Research work done by "Byung Rae Cha and Dong Seob Lee" in year 2007 in their above title paper presents, Network-based anomaly intrusion detection method using Bayesian Networks with estimated probability values of behaviors context based on Bayes theory and indirect relation [2] comes closest to my work. Their model focuses on anomalies with the TCP/IP packets coming into the network. They have take FTP service as an example and have done behavior profiling on FTP traffic coming to their server over network. They have tried to identify various patterns in network traffic coming to FTP service such as various FTP commands or strings in traffic to identify between regular and irregular traffic patterns. Their model detects irregular traffic pattern as anomaly, which eventually is marked as an attempt to break-in by their so called IDS system. They have shown various dispersion graphs and probabilities for Bayesian calculation in their paper. In their model each server which is running a service is acting as an IDS or so called HIDS. In their paper they have also tried to take the identified traffic pattern for FTP service to a NIDS, so that NIDS can detect attacks and can alert on seeing such traffic pattern(s) on network and they finish on saying that it can be done on other services too.

In my working network IIDS model probabilistic values were set to 0.5 initially for all rule sets. I'm not focusing on any specific service(s) as the above model has focused on to look at traffic patterns to learn and detect intrusion. In my model Bayesian calculations based on the probabilities are done by triggered Snort alerts and other parameters such as number of times an alert is triggered, by how many different source(s), what is the priority of the rule which has generated this alert etc, to calculate standard deviation, and get to dispersion graphs, which is basically analysis of network traffic or pattern.

I certainly feel their model can be very useful and can have better performance when compared to my model due to lack of Snort IDS rules, lack of time required to manage or update Snort IDS rules, and lack of time required to manage assigned probabilities to each Snort category/classification for Bayesian priorities calculations in my model. But, creating such a working model based on above paper will be impossible as identifying traffic pattern(s) and then analyzing them for anomalies for each service will be very resource intensive and will take lot of time to complete. And with ever changing Software development and IT infrastructure environment their model will not survive.

4.1.3. Bayesian Event Classification for Intrusion Detection

Research work done by "Christopher Kruegel, Darren Mutz, William Robertson and Fredrik Vaeur" in their above titled paper come very close to my working Bayesian model on IIDS. The key to their model is, "Instead of calculating the sum of individual model outputs and comparing the result to a threshold, they utilize a Bayesian decision process to classify input events. This process allows them to seamlessly incorporate available additional information into the detection decision and to aggregate different model outputs in a more meaningful way" [3]. But, their work is similar to work done in model [1]. They focus on application processes to identify what is a normal behavior and what is abnormal behavior with different inputs of requests and based on that calculated probability. If a new input is received which does not fall under calculated probability is flaged as an alert.

In my model, I'm taking input from current existing Snort model and comparing the results to a set threshold value which utilizes Bayesian decision tree process to classify input events based on certain pre defined probability as either True or False positive event.

I certainly feel their model is superior and can be very useful compared to my working Snort IIDS model, but practical implementation of such model is impossible as determining behavior of each process and then analyzing normal and abnormal behavior is very tedious and cumbersome task.

4.1.4. Managing Alerts in a Multi-Intrusion Detection Environment

Research work done by "Frederic Cuppens" in his above titled paper is exciting and presents a different way to manage alerts coming from multiple types of IDS such as one based on behavioral IDS model, one based on signatures, and third one on scenario recognization. Bayesian concept used in behavioral based IDS is similar to above mentioned papers, but his research goes beyond one type of IDS. In his paper Cuppens [5], talks about doing Bayesian networks based correlation of alerts coming from these three different types of IDS. Theory wise this concept sounds fascinating and this is part of ongoing project called Mirador [5], but as it is a still ongoing project there remains many issues which are still needed to be addressed, such as coming up with common form of data set from multiple types of IDS for Bayesian network to understand and correlate on various types of alerts eventually cutting down on False positives.

Cuppens [5] work maybe the right direction towards the future, where various alerts or events can be centralized into some kind of data set(s) which can be understood by one correlation engine using methods like Bayesian networks to help reduce False positives.

In my approach I'm using Bayesian decision tree to get the information from Snort alerts and correlate based on the previous selection done by system administrator as either True or False positive to identify whether the new alert with same data sets falls under same alert type previously identified as either True or False positive or as a new alert with different data sets. I feel my approach is more practical and it eventually cuts down on alerts displayed to system administrator through a web browser.

4.2. My Contribution with this Work to IDS Field

Research work done by me to present Bayesian model on working IDS successfully, addition of new code along with modification and customization of existing code is one of my major contributions in the field of Intrusion Detection System. I added new documentation and modified existing Snort documentation [12] along with the above I added steps to configure, integrate and install various Open Source packages. All the above is an important contribution to the field of Intrusion Detection System in Computer Science. There is no such complete documentation on the following important components and critical issues on Intrusion Detection Systems beside some coming from Snort project itself [13].

Some of the highlights of this project are:

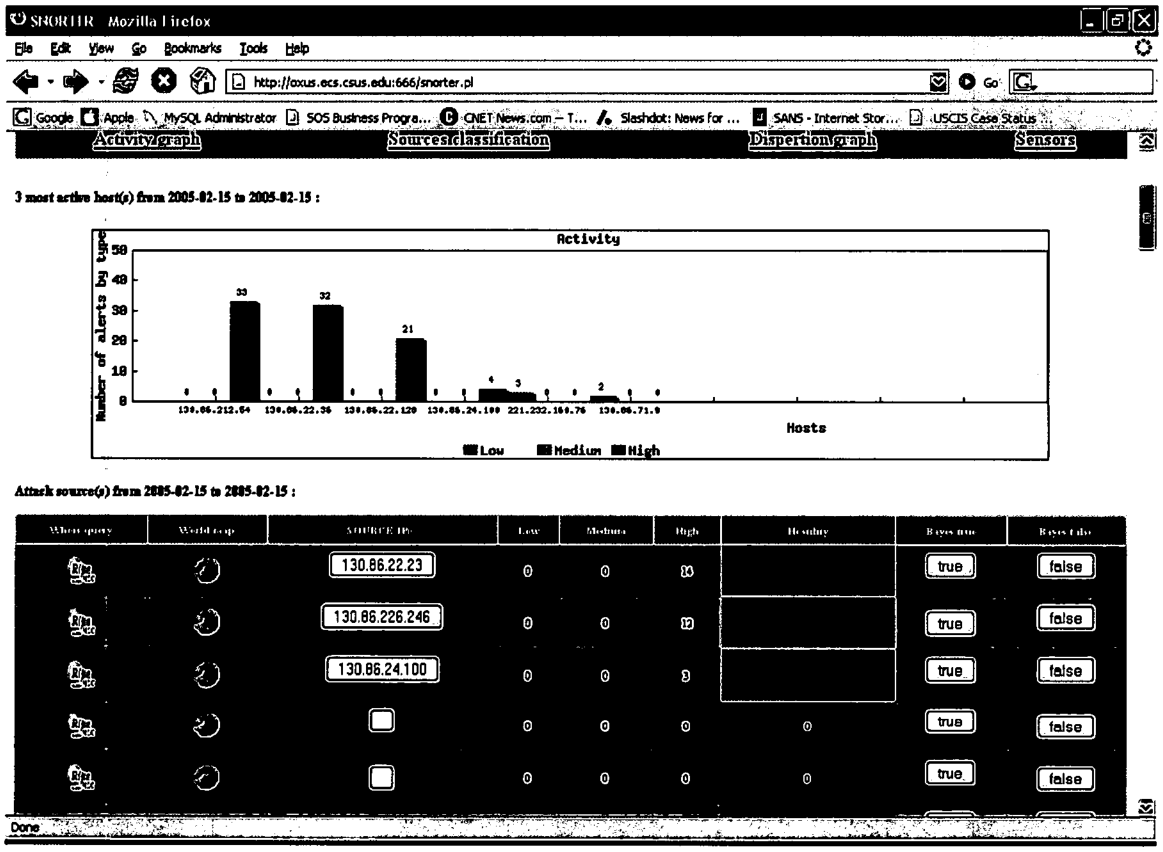
- How to install Snort on Red Hat Linux 4.0 Enterprise class server.
- How to integrate various open source software and scripts with Snort to make them successfully work as an Intrusion Detection System.
- Practical implantation using Bayesian Algorithm in Intrusion Detection Systems to cut down False positives.

Information provided in this document is easy to follow and implement. Above that, this information can be easily modified to incorporate more features. Some of the features, which can be incorporated in this project to make it a production IIDS at an enterprise class level are mentioned in Chapter 5.

4.3. Description of Output

Following is the output obtained while browsing the implemented IIDS project website in test environment. After being deployed, the application's web-module represents the application layer of the project. The website component is the sole presentation component, on which users can log in and select various queries from the browser itself to obtain multiple outputs at a time.

Figure 4.1 shows the detailed view of the application at work on a web page, when opened using client web browser. This figure shows the details of Bayesian mechanism working on the application server database and presenting alerts to a client side system. Bayesian program is written in Perl and it connects to MySQL database using Perl DBI connector and queries Snort tables which stores captured alerts from Snort engine in MySQL database. These alerts along with their probability of occurrence are displayed on the browser. Manually, just like spam e-mails user has to decide whether a shown attack is a True or a False positive attack.


SNORTER - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://oxus.ecs.csus.edu:666/snorter.pl
Google Apple MySQL Administrator SOS Business Progra... CNET News.com – T... Slashdot: News for ... SANS - Internet Stor... USCIS Case Status
Activity graph
Sources classification
Dispertion graph
Sensors
Activity
Number of alerts by type
Hosts
Low
Medium
High
Done

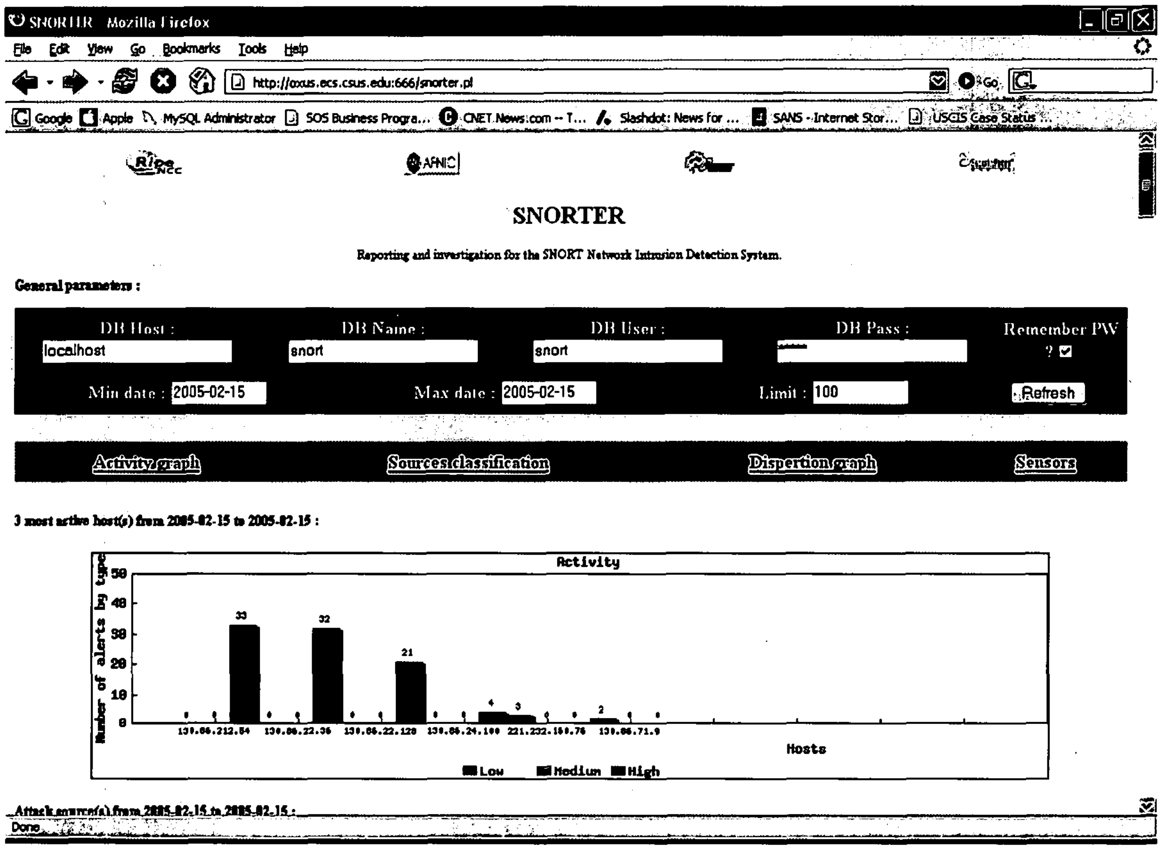
3 most active host(s) from 2005-02-15 to 2005-02-15 :

Attack source(s) from 2005-02-15 to 2005-02-15 :

Whois query	World map	SOURCE IPs	Low	Medium	High	Hostname	Bayes true	Bayes false
		130.86.22.23	0	0	18		true	false
		130.86.226.246	0	0	12		true	false
		130.86.24.100	0	0	3		true	false
			0	0	0	0	true	false
			0	0	0	0	true	false

Figure 4.1 Bayesian Working Mechanisms on Application Server

Figure 4.2 shows the detailed activity graph of attackers IP addresses. Perl program obtains IP addresses and number of alerts generated by Snort rules stored in MySQL database and then it displays them in a graphical representation. This activity can be viewed for different times by selecting different minimum and maximum dates. Internal to the program, the minimum and maximum queries executes using Perl DBI module with coded SQL statements stored in Perl program to provide an activity graph of various attackers at different time.


SNORTER - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://oxus.ecs.csus.edu:666/snorter.pl
Google Apple MySQL Administrator SOS Business Progra... CNET News.com -- T... Slashdot: News for ... SANS - Internet Stor... USCIS Case Status ...
Activity
Number of alerts by type
50
40
30
20
10
0
33
32
21
4
3
2
130.86.212.54
130.86.22.36
130.86.22.128
130.86.24.100
221.232.150.76
130.86.71.9
Hosts
Low
Medium
High
Done

SNORTER

Reporting and investigation for the SNORT Network Intrusion Detection System.

General parameters :

DB Host : localhost
DB Name : snort
DB User : snort
DB Pass :
Remember PW ? ✓
Min date : 2005-02-15
Max date : 2005-02-15
Limit : 100
Refresh

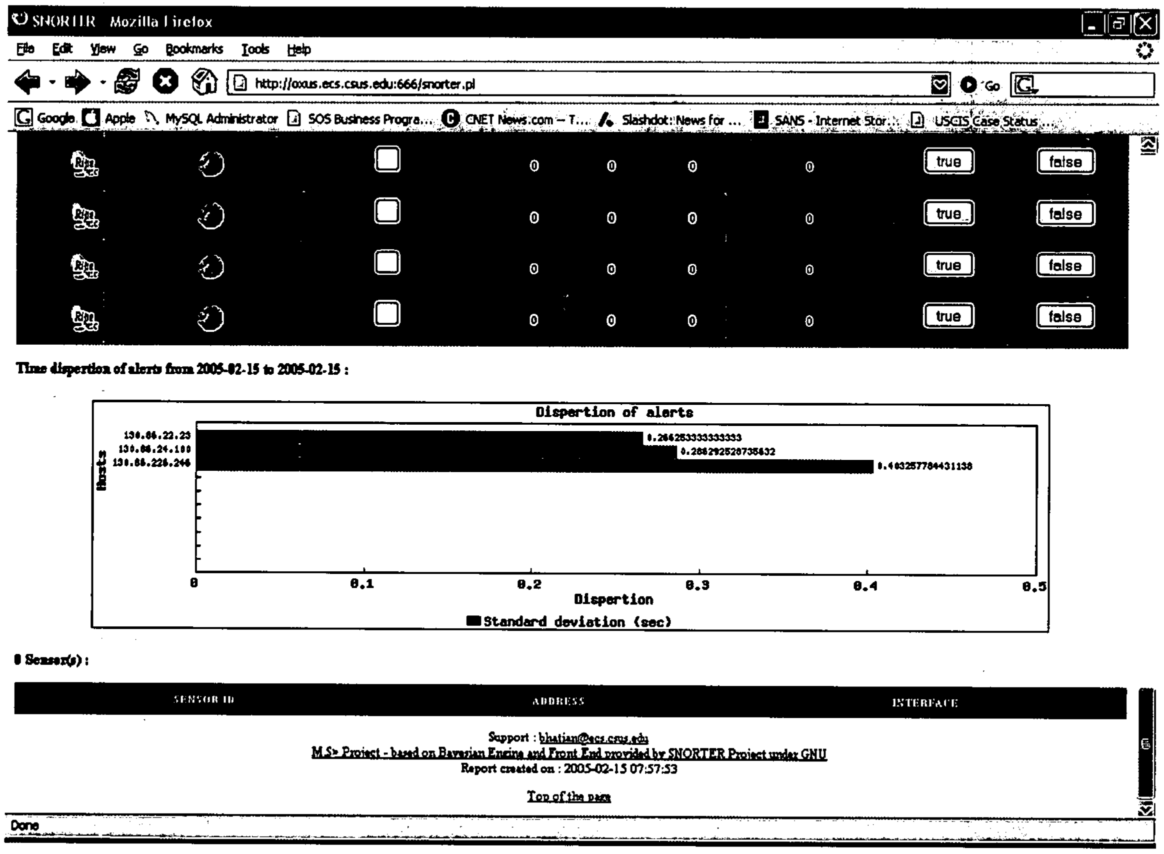
Activity graph | Sources classification | Dispertion graph | Sensors

3 most active host(s) from 2005-02-15 to 2005-02-15 :

Attack source(s) from 2005-02-15 to 2005-02-15 :

Figure 4.2 Login and Activity Graph of Attackers IP Addresses

Figure 4.3 shows the detailed dispersion graph of attackers IP addresses. Dispersion graph information is obtained by calculating the standard deviation from the different type of attacks generated by a particular attacker after obtaining the stored information in the MySQL database. Standard Deviation (SD) is calculated based on various attacks done of various levels, such as high, medium and low by a particular attack IP address. The SD calculation starts from the time since IIDS system came online and Snort started sniffing packets on network, which were stored in MySQL database. Statistical approach is used here to identify an attacker doing different attacks at various times, thinking attackers activities will not be noticed by a dedicated IDS system like Snort or firewalls which have a low buffer space to store fragmentation attacks from attackers over a period of time.



Time dispertion of alerts from 2005-02-15 to 2005-02-15 :

0 Sensor(s) :

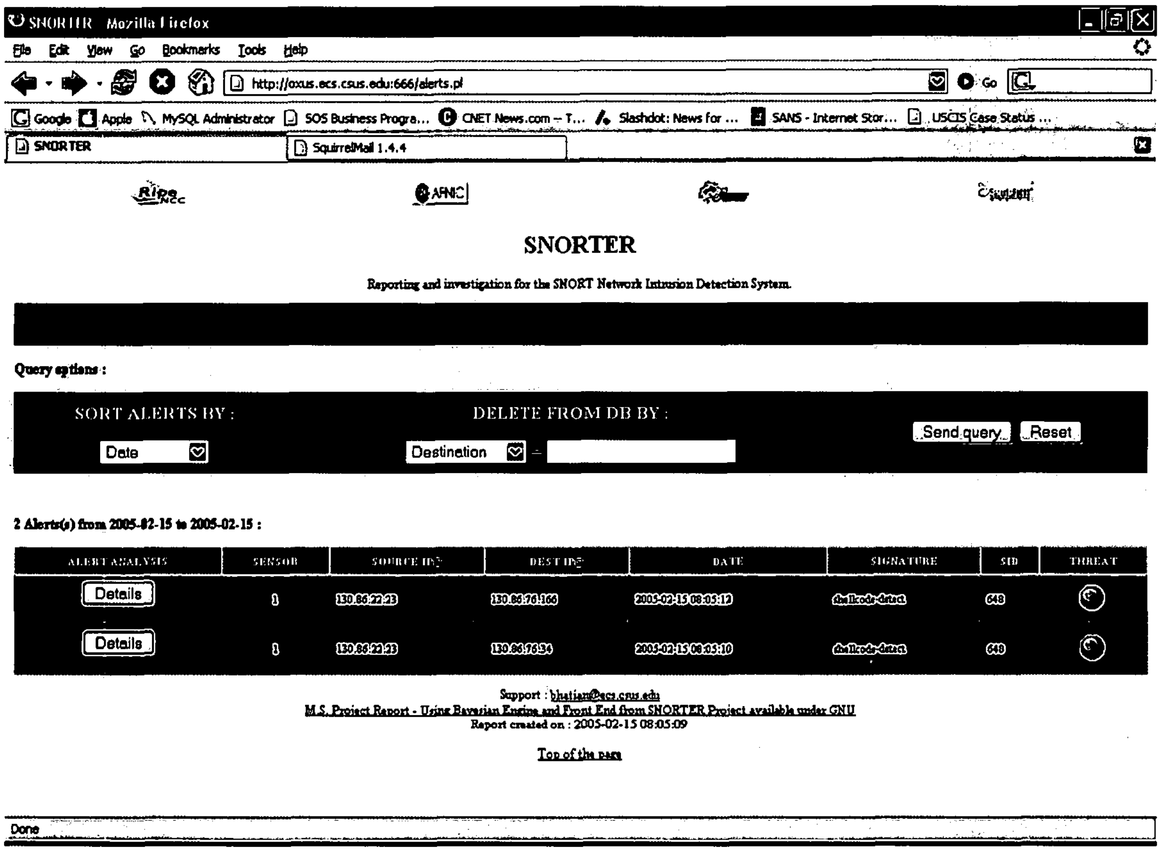
Support : bhatian@ecs.csus.edu
M.S> Project - based on Bayesian Engine and Front End provided by SNORTER Project under GNU
Report created on : 2005-02-15 07:57:53

Figure 4.3 Dispersion using Standard Deviation of Alerts

Studies have shown that a dedicated attacker first, scans a network to identify vulnerable host and then executes various attacks on vulnerable host to gain foothold in those system [5]. To come up with such an approach to calculate SD from fragmentation attacks, all customized and original Snort rules were given a 0.5 probability in the Snort Program. And depending on how many times a rule is matched from any of 'Snort Rules Groups', Standard Deviation of most active IP addresses was calculated and dispersion graph was generated for display. The mathematical formula to calculate Standard Deviation is written in Perl program, which takes input using SQL statements from MySQL database to identify most active attackers IP addresses.

Figure 4.4 shows various threat levels by giving them a visual appearance by using different colors scheme (defined below) along with attackers IP addresses and the Snort rules names evoked by that particular attacker. The various Threat level colors are Red, Blue, Orange and Yellow. Where Red identifies critical attack with high or significant chances of success or impact, Blue identifies medium attack with low chances of success, Orange identifies a lower level attack with no chances of big impact and Yellow identifies a neutral attack, which may or may not be true attack. This color pattern is generated based on the classification of Snort rules as per table 2.1. And as per entire IIDS model these alerts need to be looked at and accordingly either classification categories or Snort rules should be tuned.

NOTE: Any alert can be a critical alert, depending upon the system or resource under attack. All the important systems IP addresses should be added in "Snort.conf" file to further cut down on False positives alerts.



SNORTER

Reporting and investigation for the SNORT Network Intrusion Detection System.

Query options :

SORT ALERTS BY : Date

DELETE FROM DB BY : Destination

2 Alerts(s) from 2005-02-15 to 2005-02-15 :

ALERT ANALYSIS	SENSOR	SOURCE IP	DEST IP	DATE	SIGNATURE	SID	THREAT
Details	1	130.86.22.23	130.86.76.166	2005-02-15 08:05:12	shellcode-detect	648	
Details	1	130.86.22.23	130.86.76.34	2005-02-15 08:05:10	shellcode-detect	648	

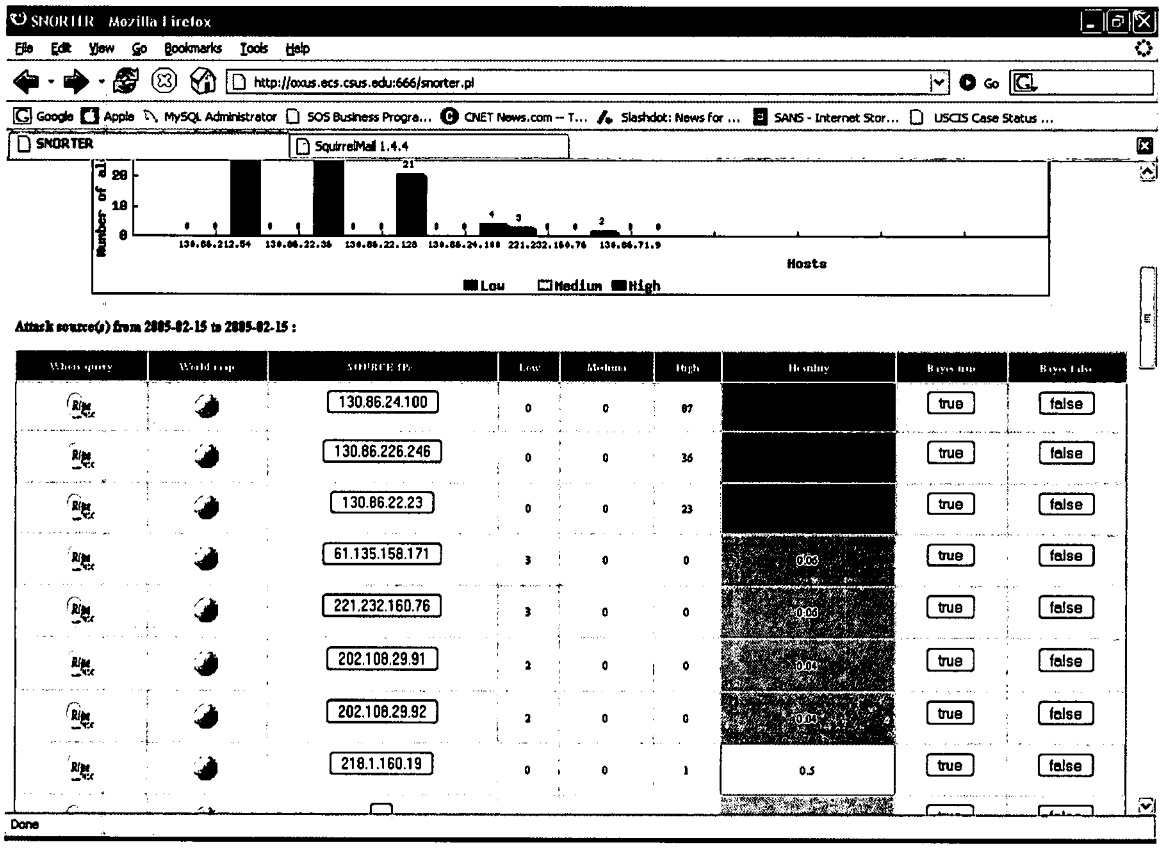
Support : bhatian@ecs.csus.edu
M.S. Project Report - Using Bayesian Engine and Front End from SNORTER Project available under GNU
Report created on : 2005-02-15 08:05:09

Top of the page

Figure 4.4 Showing Threat Levels along with Snort Signature

As mentioned previously, all major Snort rules and rules groups as per different categories are given a probability, which can be changed based on the facility where IIDS is implemented. Under Snort almost fifty 'Rules Groups/Categories' exist containing approximately twenty-six hundred rules. All these rules under Snort have different priorities which further can be modified to low or high based on the requirements of the facility where IIDS system is placed for monitoring.

Figure 4.5 shows the detailed threat level graph of attackers IP addresses, along with generated alerts. A Bayesian True and False can be selected for an individual attacker IP address making Bayesian learn whether the attacker IP address for that particular attack should be stored as a True or a False attack. If selected True here, next time attack will be flashed on web page to the user, otherwise it will not be flashed on the screen. This is how IIDS cuts down on the display of False positives or unwanted information accomplishing the main goal of this project.


SNORTER - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://oxus.ecs.csus.edu:666/snorter.pl
Go
Google Apple MySQL Administrator SOS Business Progra... CNET News.com -- T... Slashdot: News for ... SANS - Internet Stor... USCIS Case Status ...
SNORTER
SquirrelMail 1.4.4
Number of al
20
10
0
21
4
3
2
130.86.212.54
130.86.22.36
130.86.22.128
130.86.24.100
221.232.160.76
130.86.71.9
Hosts
Low
Medium
High
Done

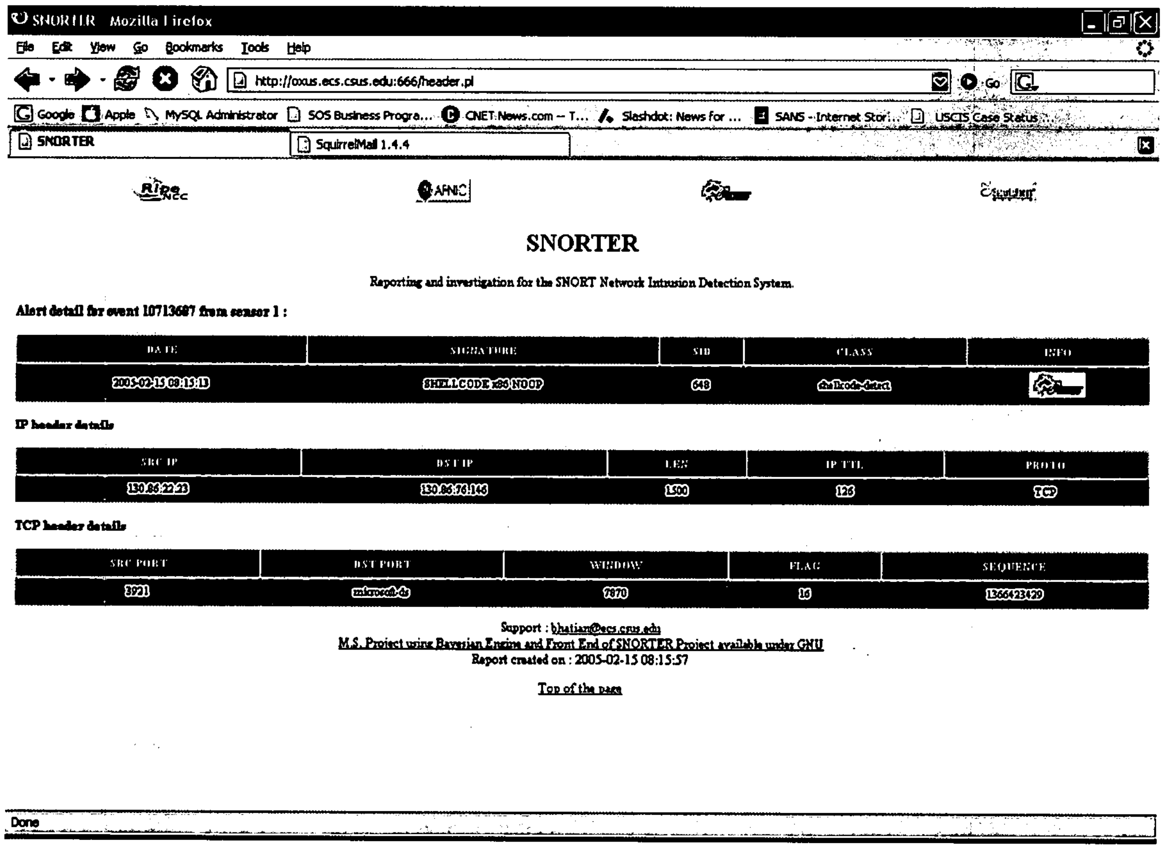
Attack source(s) from 2005-02-15 to 2005-02-15 :

Whois query	World map	SOURCE IPs	Low	Medium	High	Hostility	Bayes true	Bayes false
		130.86.24.100	0	0	87		true	false
		130.86.226.246	0	0	36		true	false
		130.86.22.23	0	0	23		true	false
		61.135.158.171	3	0	0	0.06	true	false
		221.232.160.76	3	0	0	0.06	true	false
		202.108.29.91	2	0	0	0.04	true	false
		202.108.29.92	2	0	0	0.04	true	false
		218.1.160.19	0	0	1	0.5	true	false

Figure 4.5 Showing Various Threat Levels from different Attackers IP

Figure 4.6 shows the detailed header view of the attacker IP address, along with the evoked rule from Snort rules. This can be displayed by clicking 'Details'" button in front of the attacker IP address, which is also shown in Figure 4.4. After pressing 'Details' button the information is displayed on the web browser Internal to the system, information is obtained from MySQL database after executing a SQL query written in

Perl program. The details provided are IP Header and TCP header which includes information such as Source port, Destination port along with their IP address, name of the rule evoked, Snort ID of the rule evoked, and the Class name of the Snort rule evoked. And one the best feature of this project is, if Snort rules individually in Snort



SNORTER - Mozilla Firefox

http://oxus.ecs.csus.edu:666/header.pl

SNORTER

Reporting and investigation for the SNORT Network Intrusion Detection System.

Alert detail for event 10713687 from sensor 1 :

DATE	SIGNATURE	SID	CLASS	INFO
2005-02-15 08:15:13	SHELLCODE x86 NOOP	648	shellcode-detect	

IP header details

SRC IP	DST IP	LEN	IP TTL	PROTO
130.86.22.23	130.86.76.146	1500	126	TCP

TCP header details

SRC PORT	DST PORT	WINDOW	FLAG	SEQUENCE
3921	microsoft-ds	7870	16	1366423429

Support : bhatian@ecs.csus.edu
M.S. Project using Bayesian Engine and Front End of SNORTER Project available under GNU
Report created on : 2005-02-15 08:15:57

Top of the page

File Edit View Go Bookmarks Tools Help
Google Apple MySQL Administrator SOS Business Progra... CNET News.com – T... Slashdot: News for ... SANS - Internet Stor... USCIS Case Status
SNORTER
SquirrelMail 1.4.4
Done

Figure 4.6 Showing Description of Snort Signature with Attackers IP

categories are modified then that information is automatically displayed on the web browser of the user, this is due to dynamic CGI which obtains data through a SQL query from MySQL Snort table by using Perl-DBI module. There is no need to go and modify the Perl program, other than setting up new probability values or addition of new Classes in probability file of the Snort Perl program.

Figure 4.7 shows the performance comparison between Snort IDS & IIDS alerts. This graph is generated based on seven days of comparison of alerts between two systems, normal IDS based and configured as per references [10,12,13,14] and an IIDS. Table 4.1 provides the data set which has been used to plot this graph in terms of alerts produced by Snort based IDS and alerts produced by Bayesian based IIDS with respect to time interval of eight hours for period of seven days.

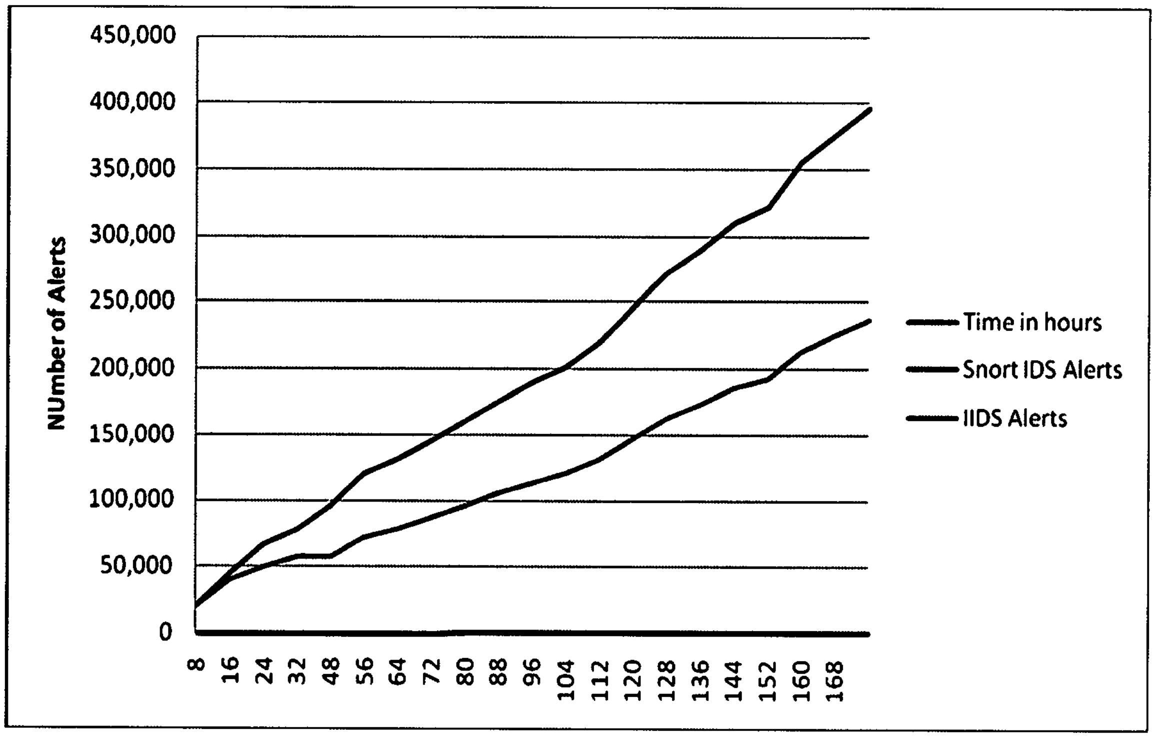


Figure 4.7 Showing Performance Comparison Between Snort IDS alerts and IIDS alerts in period of Seven Days.

The main difference between these two alert lines is based on the number of False positives alerts which when selected as False by system administrator, these False positives alerts are removed from IIDS alerts when displayed on a web browser. Hence, making it easy for a system administrator to browse through the alerts and manage new alerts easily.

NOTE: In both systems new rules were added and outdated rules were purged. In IIDS after detecting an alert as False positive rules were customized or disabled further to cut down such alerts, where as there was no fine tuning of rules done on normal Snort IDS.

Time (in hours)	IDS Alerts	IIDS Alerts
8	20000	20000
16	45000	40000
24	66000	50000
32	78000	58000
40	96000	57600
48	120000	72000
56	132000	79200
64	146000	87600
72	160000	96000
80	176000	105600
88	190000	114000
96	202000	121200
104	220000	132000
112	246000	147600
120	272000	163200
128	290000	174000
136	310000	186000
144	322000	193200
152	355000	213000
160	376000	225600
168	396000	237600

Table 4.1 Comparison of Number of Alerts in IDS and IIDS

Chapter 5

CONCLUSION

The completion of Intelligent Intrusion Detection System project has been a very satisfying experience. The initial learning curve of this project was steep in the beginning as I did not have much experience in developing Perl and MySQL integration project at enterprise level. Learning new technologies such as integrating Perl modules, writing Perl packages, integrating Open Source software with Perl, writing CGI, HTML interfaces, SQL statements embedded in Perl was initially time consuming as well as demanded serious hard work. Writing set of customized applications with backend database and setting up all application processes running on Linux server along with writing automated jobs that are easily configurable to run in tandem were also a cumbersome task. Once I got over with my learning curve and found my favorite tools, from then on it was all challenging programming that I have always cherished throughout my student life. The ability to create things and see their output materialize in seconds is an immense feeling.

Meetings with faculty advisor Dr. Du Zhang in the initial phase of project proved very helpful especially towards directing me to Bayesian algorithm implementation. Meeting with Prof. Richard Smith led to the conception of this project and helped me identify the challenges which lies in current Snort based Intrusion Detection Systems. Both Prof. Richard Smith's and Dr Du. Zhang's timely help lead to the successful conclusion of the project. The manual learning mechanism of Bayesian algorithm for True attacker and cutting down of the number False positive attacks was the main goal of this project. The biggest advantage with IIDS project is, that with reduced number of alerts due to decrease in False positive alerts it has decreased manual time consumption of going through alerts on normal Snort based IDS.

5.1. Current Drawbacks Associated with this Project:

Every system has some drawbacks. IIDS system has a major drawback, which makes it to stale after processing True and False positives attacks after certain number of alerts processing.

- Performance Issues: These are mainly due to limited memory capacity in current test hardware system and Snort processing will stop after running for sometime. This is due to program implementation in Perl using hashes. Hash sizes in Perl can increase to a fix size based on the available system memory. In this project hashes maintain the list of attackers IP's who are generating both False and True attacks, and by having hashes I was able to cut down on False positive alerts quickly. There is a huge memory requirement based on the size of hash table, especially to do the comparison of both the True and False attacker's hashes running on same server which is running MySQL and Snort IDS processes. As hash table continues to grow in size the performance of system decreases exponentially. One way this can be reduce or decreased is by having specific and only real alert generating Snort rules. This will require manually fine tuning of

Snort rules and groups before brining it online and passing the information to IIDS.

- One of the biggest drawbacks with any IDS system is distributed denial of service attacks also called DDOS [12]. Due to limited physical memory and ever growing hash alert table IIDS type of a system is prone to Denial of Service Attack. After doing a DDOS attack on IIDS kind of a system which will stale the IIDS system, an attacker can continue to target on that enterprise or business which is using this kind of a system.

- IIDS system does require fine tuning of Snort rules very frequently more than a normal Snort based IDS. Fine tuning of Snort rules is tedious and time consuming, but eventually it helps to cut down on False positive alerts.

5.2. Future Work or Modifications:

Any future attempt to further extend this project should not face any problems in understanding code and other integration and implementation work. The coding of this project is done in a lucid way with complete comments in the beginning of modules.

Following future work/additions/modifications can be done on this project to incorporate other advanced features that I could not work on; given the dearth of time:

- Currently, using this project True attackers IP addresses can be extracted to a text file from database with some SQL statements. A simple parser can be written that essentially will extract all the True attackers IP addresses from a MySQL database using SQL statements and output in the form of Access Control List's (ACL's) in a text file. Both In-line Linux firewall and an Enterprise Cisco Firewall can be configured to drop IP addresses requests using the obtained text file of the True attackers IP Addresses. One way to do that is manually update both In-line Firewall and enterprise Cisco Firewall to block new attackers IP addresses. This manual task can be further automated but, it requires strong understanding of network and domains across the network(s).

- Paging and Cell Phone messaging feature can be added to alert Network and System administrators. If a particular attack is critical or an attack has been executed at a specific number of times on an internal business or critical server or on an application then such alerts can be sent out to administrators. This feature can be added as a cron job. Paging software is available as open source software for Linux platform. Scripting in Perl is required to integrate this available software with MySQL database and this can be included with the Perl code too. Some good planning and understanding of complete application is required here to make this possible.

APPENDIX A

Installation Manual

1. Requirements before Installation

IP address for the network card which will be assigned for management of IIDS. We also need to know domain name servers also called DNS server IP address and name, which will be used for resolving hostnames to IP's and for managing the IIDS by hostname.

2. Installing Red Hat Enterprise Edition

We will install a minimal number of packages sufficient for a usable system. After the install we'll turn off anything that is not needed. It is an ideal dedicated IIDS by hardening the OS and further securing the system.

Under Network Configuration

Select edit and uncheck "Configure with DHCP" for the network card, which will be used for sniffing network traffic. Do no uncheck the network card, which will be used for monitoring the IIDS.

NOTE: If you have more than one network card sniffing on various networks, uncheck eth1 etc if you have more network interface cards also called "NIC" in short. Do not uncheck network card (eth0), which should be the monitoring network card.

Leave "Activate on boot." Set a static IP and subnet mask for your network and manually set the hostname for the system. Then set a gateway and DNS addresses.

NOTE: Always try to assign a static IP address here. I think it is best not to run Snort off of a Dynamic IP, however, if you need to, go ahead and do it, just make sure to point your $HOME_NET variable in your snort.conf to the interface name. You can get more info on that in the Snort FAQ. If this is a dedicated IDS then you do not need to have an IP on the interface that Snort is sniffing.

Firewall

Choose “enable firewall”

Select remote login (SSH) and Web Server (HTTP, HTTPS)

For the SELinux option, move to Disabled.

3. Suggested Packages to Install

Take the defaults with the following exceptions. (Default is whatever it has when you choose custom; for example, Gnome is checked by default and Kde is not.)

Desktops

- X Window System – click “details” and uncheck the following modules:
- “xisdnload” (unless you are going to be directly connected to an ISDN line.)
- VNC Server
- “Gnome Desktop Environment” – Accept the default (checked.)
- “KDE Desktop Environment” – Check to select it.

Applications

- “Editors” – Choose your favorites.
- “Engineering and Scientific” – Accept the default (unchecked.)
- “Graphical Internet” – Select this one and click “details” and install only the following:
- Firefox web browser.
- “Text based Internet” – Select this one and click “details” and install only the following:
- Lynx – a text based web browser.
- Pine – a text based e- mail client.
- “Office/Productivity” – Select this one and click “details” and install only the following:
- gpdf
- Openoffice.org
- “Sound and Video” – None of this is needed.
- “Authoring and Publishing’ – None of this is needed.
- “Graphics” – Select this one and click “details”. Check the following:
- Gimp – Good to have if you are using Gnome.
- Gimp data extras.
- Gimp print plugin.

- "Games and Entertainment" – None of this is needed.

Server Section

- Choose nothing from this entire section – Check and leave at the default
- Web Server – Only the following should be checked
- Crypto-Utils
- Mod_auth_mysql
- Mod_perl
- Mod_ssl
- Php
- Php_mysql
- System-config-httpd
- Webalizer
- Mail Server – None of this is needed.
- Windows File Server – None of this is needed.
- DNS Server – None of this is needed.
- FTP Server – None of this is needed.
- Postgresql Database –None of this is needed.
- MySQL Database – Check only the following
- MyODBC
- Mod_auth_mysql

- Mysql-devel
- Mysql-server
- Mysqlclient 10
- Perl-DBD-MySQL
- Php-mysql
- News Server – None of this is needed.
- Network Servers – None of this is needed.
- Legacy Network Servers – None of this is needed.

Development

- “Development Tools” – Select this one and click “details” and check the following in addition to what is checked by default.
- Expect
- Gcc-objc
- “Kernel Development” – Everything is selected by default.
- “X Software Development” – Leave this unchecked.
- “Gnome Software Development” – Leave this unchecked.
- “KDE Software Development” – Leave this unchecked.

System

- “Administration” – check and accept default.

- “System Tools” – Select this one and click “details” and check only the following (some will required to be unchecked.)
- Ethereal
- Ethereal gnome
- Nmap
- Nmap front-end
- “Printing Support” – Uncheck this (unless you need printing from this machine, then configure as needed.)

Miscellaneous

Choose nothing from this entire section.

And boot the system.

User Account

Add a user account for yourself here and make sure to set up a strong password.

The root account should not be used for everyday use, if you need access to root functions then you can “su -“ or “sudo” for root access.

Red Hat Network

Register your system with RHN here so that you will be notified when new patches come out. By using the command (#rhn_register) as a root.

Login to the system (unless you are coming here from the previous section)

Now we need to disable services that you will not need for this system. First, login as root. Then click on the Red Hat on the bottom/top left of the toolbar. Select System Settings, then Server Settings, then Services. This will bring up the list of services that start when the system boots up. Disable the following services:

apmd, cups, first boot, isdn, netfs, nfslock, pcmcia, portmap, sgi_fam.

Click on "Save" at the top of that window, and close the service configuration.

Securing SSH

In the /etc/ssh/sshd_config file change the following lines (if it is commented out, remove the #):

```
Protocol 2
PermitRootLogin "no"
PermitEmptyPasswords "no"
```

Update and reboot your system (you might have installed a new kernel, when you updated the installation and changed the SSH config, so a reboot is necessary). You are now up to date with all the latest packages and you can start the Snort install.

Download all the needed files:

You are now ready to start installing Snort and all the software it needs. You can either use the desktop terminal window, or SSH into the server from another box, either way will work fine.

Place all the downloaded files as shown in Chapter 5, into a directory for easy access and consolidation. This directory will not be needed when you are finished with the installation and may be deleted at that time. I create a directory under /root called snortinstall. Use the mkdir command from the shell. Make sure you are in the /root directory (cd /root). You can check where you are currently by using the pwd command.

4. Integration of various Open Source Software Applications

Preparing for the install

Again, if you are not logged in as root, then you will need to "su –" to root. Please ensure that you have downloaded all of the installation files before you start the install. Go to your download directory and start with the following procedures. They will walk you through extracting the source files of the applications, compiling, then installing and configuring them for use with Snort.

NOTE: Remember, if you use SSH to access the box, you will need to use the regular user account you created when you installed Red Hat. Then "su –" to the root account.

Links for Open Source Software download

When you are using a Terminal Window, or, if you are SSH into the box, you can use wget (wget will place the file you're downloading into the directory where you're currently located) to download these files. To use wget, type "wget <URL_to_file>", and it will begin the download to the directory that you are currently in.

NOTE: Download the Following or their latest stable releases:

Download Snort 2.3.3

http://www.snort.org/dl/current/snort-2.3.3.tar.gz

Download ADODB 462

http://easynews.dl.sourceforge.net/sourceforge/adodb/adodb462.tgz

Download JPGraph 1.19

http://members.chello.se/jpgraph/jpgdownloads/jpgraph-1.19.tar.gz

Download LibPcap 0.8.1

http://umn.dl.sourceforge.net/sourceforge/libpcap/libpcap-0.8.1.tar.gz

Download LibPng

http://umn.dl.sourceforge.net/sourceforge/libpng/libpng-1.2.8-config.tar.gz

Download PCRE

http://easynews.dl.sourceforge.net/sourceforge/pcre/pcre-5.0.tar.gz

Download DBI

http://search.cpan.org/CPAN/authors/id/T/TI/TIMB/DBI-1.48.tar.gz

Download CGI.pm

http://search.cpan.org/CPAN/authors/id/L/LD/LDS/CGI.pm-3.11.tar.gz

Download gd

http://www.boutell.com/gd/http/gd-2.0.33.tar.gz

Download Freetype

http://savannah.nongnu.org/download/freetype/freetype-2.1.10.tar.gz

Download Jpeg

http://www.ijg.org/files/jpegsrc.v6b.tar.gz

Download Xpm

http://koala.ilog.fr/ftp/pub/xpm/xpm-3.4k.tar.gz

Download GD-2.28

http://search.cpan.org/CPAN/authors/id/L/LD/LDS/GD-2.28.tar.gz

Download GDGraph

http://search.cpan.org/CPAN/authors/id/M/MV/MVERB/GDGraph-1.43.tar.gz

Download GDTextUtil

http://search.cpan.org/CPAN/authors/id/M/MV/MVERB/GDTextUtil-0.86.tar.gz

Download S99snort Bash Script

http://cvs.snort.org/viewcvs.cgi/snort/contrib/Attic/S99snort?rev=1.2

5. Install LibPcap

```
tar xvzf libpcap-0.8.1.tar.gz

cd libpcap-0.8.1

./configure

make

make install; cd ..
```

6. Install LibPng

```
tar xvzf libpng-1.2.8.tar.gz

cd libpng-1.2.8

./configure
```

```
make

make install; cd ..
```

7. Install PCRE

```
tar xvzf.  pcre-5.0.tar.gz

cd pcre-5.0

./configure

make

make install; cd ..
```

8. Install DBI

```
tar xvzf DBI-1.48.tar.gz

cd DBI-1.48

perl Makefile.PL

make; make test

make install ; cd ..
```

9. Install CGI.pm

```
tar xvzf  CGI.pm-3.11.tar.gz

cd CGI.pm-3.11

perl Makefile.PL

make; make test
```

```
make install ; cd ..
```

10. Install gd

```
tar xvzf gd-2.0.33.tar.gz
cd gd-2.0.33
./configure
make
make install; cd ..
```

11. Install Freetype

```
tar xvzf  freetype-2.1.10.tar.gz
cd freetype-2.1.10
./configure
make
make install; cd ..
```

12. Install Jpeg

```
tar xvzf  jpegsrc.v6b.tar.gz
cd jpeg-6b
./configure
make
```

NOTE: Open Makefile in your favorite editor and under "Where to install the programs and man pages, add # in front of the line mandir = $(prefix)/man/man$(manext)

```
make install; cd ..
```

13. Install Xpm

```
tar xvzf  xpm-3.4k.tar.gz
cd xpm-3.4k
xmkmf -a
make
make install
make install.man; cd ..
```

14. Install GD-2.28

```
tar xvzf  GD-2.28.tar.gz
cd GD-2.28
perl Makefile.PL
make
make install; cd ..
```

15. Install GDGraph

```
tar xvzf  GDGraph-1.43.tar.gz
cd GDGraph-1.43
```

```
perl Makefile.PL
make
make install; cd ..
```

16. Install GDTextUtil

```
tar xvzf GDTextUtil-0.86.tar.gz
cd GDTextUtil-0.86
perl Makefile.PL
make
make install; cd ..
```

17. Install MySQL

Turn on and set to start service:

```
chkconfig mysqld on
service mysqld start
```

Test to see if it worked:

```
/usr/local/mysql/bin/mysqld_safe --user=mysql &
```

(you might have to hit enter to get back to a shell prompt)

If you get no errors, type "ps –ef | grep mysql". You should see something like this:

```
[root@IDS mysql-4.0.13]# ps -ef | grep mysql
root 13297 2290 0 11:20 pts/0 00:00:00 /bin/sh /usr/local/mysql/bin/mysqld_safe --
user=mysql
```

```
mysql 13319 13297 3 11:20 pts/0 00:00:00 /usr/local/mysql/libexec/mysqld --
basedir=/usr/local/mysql
--datadir=/usr/local/mysql/var --user=mysql --pid-file=/usr/local/mysql/var/nitin-
redhat.pid –skiplocking
```

Set MySQL to start automatically:

Use the following commands to create symbolic links in the startup folders for run levels 3 and 5. MySQL will now start automatically when you boot up.

```
cd /etc/rc3.d
ln -s ../init.d/mysqld S85mysqld
ln -s ../init.d/mysqld K85mysqld
cd /etc/rc5.d
ln -s ../init.d/mysqld S85mysqld
ln -s ../init.d/mysqld K85mysqld
```

NOTE: We will setup MySQL Database later in this document after installing Snort.

18. Installing and configuring Apache

Turn on and set to start service:

```
chkconfig httpd on
service httpd start
```

Check the system to make sure the web server is working (go to the IP of that system in a Web browser. You will see the default Apache page.)

Apache is now configured on this system.

Set Apache to start automatically:

```
cd /etc/rc3.d
ln -s ../init.d/httpd S85httpd
ln -s ../init.d/httpd K85httpd
cd /etc/rc5.d
ln -s ../init.d/httpd S85httpd
ln -s ../init.d/httpd K85httpd
```

NOTE: The above lines will add a start-up script to the system for both run-level 3 and 5.

19. Installing and setting up Snort and the Snort rules:

Go to the directory where Snort was downloaded and on command prompt type the following commands.

```
groupadd snort
useradd -g snort snort

mkdir /etc/snort
mkdir /var/log/snort
mkdir /etc/snort/rules
```

NOTE: Make sure the directories are created.

```
tar xvzf snort-2.3.3.tar.gz
cd snort-2.3.3
./configure --with-mysql
make
make install
```

Installing the rules and conf file:

NOTE: From the Snort installation directory. At the time of this document, Snort team was planning to separate rules from original basic Snort Installer. If they have separated rules, then download them in the Snort directory and continue with the steps below.

```
cd rules
cp * /etc/snort/rules
cd ../etc
cp *  /etc/snort
```

Modify your Snort.conf file:

The snort.conf file is located in /etc/snort, make the following changes.

NOTE: You can do as many changes you want, including adding new features in this file leading to addition of new features in Snort. But, before that make sure you know "Programming in Bash" and have a good understanding of Snort. You can also write your own rules. Also, with updated versions after this documentation, it is REQUIRED to understand all the changes done to the new version of Snort and accordingly modify Snort.conf

var HOME_NET 10.2.2.0/24 (make this whatever your internal network is!)

NOTE: Only change it, when you are using Snort to sniff your internal network, if it's in front of your firewall, leave it to any.

Change the rule path variable

var RULE_PATH /etc/snort/rules

NOTE:

1. Please do the following changes after understanding the updates and changes in the version of Snort you are installing. Also, to do the changes below it is REQUIRED that you have a good understanding of Snort. If you are not aware of Snort internal working then follow steps 2 below.
2. Jump to modify the database logging connection in this section and put the proper password.

Add the following like this:

Preprocessor bo: -nobrute

Remove Hash(#) in front of the following:

Preprocessor Conversation

Preprocessor Portscan2: (Here you can also change the values of Scanners_max and Targets_max depending upon the traffic on your network.)

Database logging connection:

Tell it to log to the database (make sure this is on one line) new_password is whatever you want as long as it is the same when you set up mysql

output database: log, mysql, user=snort password=new_password dbname=snort host=localhost sensor_name=hostname.xyz.com:eth1 (Put the name of your sniffing NIC)

NOTE: You need to add sensor_name in the same line. And add all other NIC's which you want to sniff by repeating the sensor_name=hostname.xyz.com:eth2 etc in same line.

20. Set Snort to start automatically:

Snort can be setup to come up to sniff on a designated interface in a promiscuous mode. Go to your "snortinstall" download directory and copy it to "/etc/init.d" and call it's not. By doing the following:

```
cp S99snort  /etc/init.d/snort
cd /etc/init.d
```

And add these lines in "snort" and add the following lines after "Set Interface"

```
IFACE=eth1
```

```
ifconfig  $IFACE promisc
CONFIG=/etc/snort/snort.conf
SNORT_GID=snort
```

And then modify the line like the following in the same file.

```
/usr/local/bin/snort –c /etc/snort/snort.conf –i $IFACE –g snort  –D (one line)
```

Now go to:

```
cd /etc/rc3.d
ln -s ../init.d/snort S99snort
ln -s ../init.d/snort K99snort
cd /etc/rc5.d
ln -s ../init.d/snort S99snort
ln -s ../init.d/snort K99snort
```

NOTE: Take care of the errors in snort.conf, if you have modified snort.conf. This is important, as it will give you output that will be helpful. It will tell you if you are having problems with rules or if you have a bad line in your conf file. Make sure that the line for MySQL in the snort.conf file is not wrapped or cut into two lines or if you have missed some steps in this document.

21. Setting up the database in MySQL:

I will put a line with a ">" in front of it so you will see what the output should be.

NOTE: In MySQL, a semi-colon " ; "character is mandatory at the end of each input line.

(new_password is whatever password you want to give)

```
#mysql

mysql> SET PASSWORD FOR root@localhost=PASSWORD('new_password');

>Query OK, 0 rows affected (0.25 sec)

mysql> create database snort;

>Query OK, 1 row affected (0.01 sec)

mysql> grant INSERT,SELECT on root.* to snort@localhost;

>Query OK, 0 rows affected (0.02 sec)

mysql> SET PASSWORD FOR snort@localhost=PASSWORD('new_password');

>Query OK, 0 rows affected (0.25 sec)

mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to
snort@localhost;

>Query OK, 0 rows affected (0.02 sec)

mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort;

>Query OK, 0 rows affected (0.02 sec)

mysql> exit

>Bye
```

From the “Snort” source directory execute the following command (when working with MySQL, if it asks for a password it wants the one you defined in the SQL statement “SET PASSWORD FOR root@localhost=PASSWORD('new_password');”)

```
mysql -u root -p < ./Snort/create_mysql snort

Enter password:
```

Now you need to check and make sure that the Snort DB was created correctly

```
mysql –p

>Enter password:

mysql> SHOW DATABASES;
```

(You should see the following)

```
+------------+

| Database

+------------+

| mysql

| snort

| test

+------------+

3 rows in set (0.00 sec)

mysql> use snort


>Database changed


mysql> SHOW TABLES;
```

```
+------------------+

| Tables_in_snort

+------------------+

| data

| detail

| encoding

| event

| flags

| icmphdr

| iphdr

| opt

| protocols

| reference

| reference_system

| schema

| sensor

| services

| sig_class

| sig_reference

| signature

| tcphdr

| udphdr
```

```
+------------------+

19 rows in set (0.00 sec)

>Bye
```

To Verify Snort Installation:

Type the following on the command prompt

```
/usr/local/bin/snort –c /etc/snort/snort.conf
```

22. Install JPGraph

Go back to your downloads directory

```
cp jpgraph-1.19.tar.gz /var/www/

cd /var/www/


tar xvzf  jpgraph-1.19.tar.gz

rm –rf jpgraph-1.19.tar.gz

cd jpgraph-1.19

rm -rf README

rm -rf QPL.txt
```

23. Installing ADODB

Go back to your initial download directory

```
cp adodb462.tgz /var/www/
```

```
cd /var/www/

tar xvzf adodb462.tgz

rm –rf  adodb462.tgz
```

24. Installing and Configuring Snort

- un-tar the package and cd in SNORT directory.

Edit the makefile to check the default install and network connection options:

```
ROOT = /opt/snort

SOURCEPATH = .

CONFIGPATH = /etc/snort

BIN=snort.sh

BINPATH=/usr/sbin

certname = snort

life = 730

keylength = 1024

listenport = 666

listen_ip = 127.0.0.1

KEY_REPOSITORY = $(CONFIGPATH)/ssl_key

SSL_PROGRAM = /usr/bin/openssl
```

- Do a "make install", setup will ask you :

  - information needed to build the SSL certificate

- information needed to install Perl GD graphic library

- Start Snort in background with "snort.sh&". Write a "cron-job" to start this process whenever it dies and on every boot of system.
- Connect with your favorite brówser to the address and port specified in the Makefile (default is 127.0.0.1:666). You will be prompted for Mysql connection parameters.

Check to see if everything is working:

If you have set up your NIC the way you wanted. Reboot your system and watch to make sure everything starts. You can check by doing a "ps –ef |grep <service>" the service can be any running process. i.e. mysql, httpd, snort and snort

NOTE: If you want the machine to start at a text prompt instead of X, then change the default in the inittab file (/etc/inittab) from 5 to 3. Go to a shell as root and check everything important to see if it is running.

To check you can execute "ps –ef |grep httpd && ps –ef |grep mysql && ps –ef |grep snort" (One line)

25. PURGING Scripts

NOTE: For Linux, BSD and UNIX Systems only.

Get the scripts from here: http://gaia.ecs.csus.edu/~bhatian/purge and do the following modifications:

In Purge_database.sh, do the following:

Change the AGE

NOTE: 1. This is for the number of day's alert data you want your IIDS to show. Remaining it will purge automatically if configured using cron job. But before running this from cron, take care of your MySQL-Socket now, if you have missed that step previously. As, this cron job will delete MySQL-Socket in temp directory if it is not set to sticky bits. Once, the socket is deleted, you have to go into MySQL DBMS and drop complete database and all the data inside them. Finally, Configuring and Installing MySQL and repeating Setting up Database in MySQL process as mentioned above.

AGE=4:00:00

Use your 'new_passowrd', which you used in your Snort database.

It should look like this:

DATABASE=snort

USER=snort

PASSWORD=new_password

26. Setting up NIC's Parameters in Linux

Go to "/etc/sysconfig/network-scripts"

You will see ifcfg-eth0, ifcfg-eth1 etc.

Using your favorite editor. Open the ifcfg-eth1

NOTE: Assuming, eth1 is your sniffing NIC. And, if you have more than one sniffing NIC's do the same in all of them.

Set the following. Some of them will be already set, so do not change them.

```
DEVICE=eth1
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
PEERDNS=no
TYPE=Ethernet
```

Do the above to all the NIC's, and then reboot.

APPENDIX B

Configuration Details and Source Code

Software Configuration Details

The Operating System on which this system was developed and installed was Red Hat Linux 4.0 Enterprise Edition. The O.S. was installed with the following key components required by this project. The packages were: Libpng, PCRE, Apache and MySQL.

Hardware Configuration Details

Dell Xenon Dual processors with 3 GHz CPU, with 2 Giga Bytes of RAM, with 3 NIC each of I Giga Bytes capacity and with 36 Giga Bytes of Hard Disk having 7500rpm connected using RAID 1.

Showing Bayesian Class

```
icmp-event:0.5:0.5
misc-activity:0.5:0.5
network-scan:0.5:0.5
not-suspicious:0.5:0.5
protocol-command-decode:0.5:0.5
string-detect:0.5:0.5
unknown:0.5:0.5
bad-unknown:0.5:0.5
attempted-dos:0.5:0.5
attempted-recon:0.5:0.5
denial-of-service:0.5:0.5
misc-attack:0.5:0.5
non-standard-protocol:0.5:0.5
rpc-portmap-decode:0.5:0.5
successful-dos:0.5:0.5
successful-recon-limited:0.5:0.5
successful-recon-largescale:0.5:0.5
suspicious-filename-detect:0.5:0.5
suspicious-login:0.5:0.5
system-call-detect:0.5:0.5
unusual-client-port-connection:0.5:0.5
```

```
web-application-activity:0.5:0.5
attempted-admin:0.5:0.5
attempted-user:0.5:0.5
shellcode-detect:0.5:0.5
successful-admin:0.5:0.5
successful-user:0.5:0.5
trojan-activity:0.5:0.5
unsuccessful-user:0.5:0.5
web-application-attack:0.5:0.5
policy-violation:0.5:0.5
default-login-attempt:0.5:0.5
```

Showing Snort.pl

```
#!/usr/bin/perl
print "Content-type: text/html\n";
# GNU GENERAL PUBLIC LICENSE http://www.gnu.org/copyleft/gpl.html

use CGI;
use DBI;
use GD::Graph::bars;
use GD::Graph::hbars;
use strict;

my %weight; # this is the metric database

$weight{'icmp-event'} = "0.01";
$weight{'misc-activity'} = "0.02";
$weight{'network-scan'} = "0.03";
$weight{'not-suspicious'} = "0.001";
$weight{'protocol-command-decode'} = "0.02";
$weight{'string-detect'} = "0.01";
$weight{'unknown'} = "0.01";
$weight{'bad-unknown'} = "0.2";
$weight{'attempted-dos'} = "0.02";
$weight{'attempted-recon'} = "0.02";
$weight{'denial-of-service'} = "0.5";
$weight{'misc-attack'} = "0.5";
$weight{'non-standard-protocol'} = "0.1";
$weight{'rpc-portmap-decode'} = "0.1";
$weight{'successful-dos'} = "0.1";
$weight{'successful-recon-largescale'} = "0.1";
$weight{'successful-recon-limited'} = "0.1";
$weight{'suspicious-filename-detect'} = "0.1";
$weight{'suspicious-login'} = "0.1";
$weight{'system-call-detect'} = "0.1";
$weight{'unusual-client-port-connection'} = "0.1";
$weight{'web-application-activity'} = "0.1";
$weight{'attempted-admin'} = "0.2";
```

```
$weight{'attempted-user'} = "0.2";
$weight{'shellcode-detect'} = "1";
$weight{'successful-admin'} = "1";
$weight{'successful-user'} = "1";
$weight{'trojan-activity'} = "0.5";
$weight{'unsuccessful-user'} = "0.5";
$weight{'web-application-attack'} = "0.2";
$weight{'policy-violation'} = "1";
$weight{'default-login-attempt'} = "0.5";

my %class; # this is a simple snort severity database
$class{'icmp-event'} = "low";
$class{'misc-activity'} = "low";
$class{'network-scan'} = "low";
$class{'not-suspicious'} = "low";
$class{'protocol-command-decode'} = "low";
$class{'string-detect'} = "low";
$class{'unknown'} = "low";
$class{'bad-unknown'} = "medium";
$class{'attempted-dos'} = "medium";
$class{'attempted-recon'} = "medium";
$class{'denial-of-service'} = "medium";
$class{'misc-attack'} = "medium";
$class{'non-standard-protocol'} = "medium";
$class{'rpc-portmap-decode'} = "medium";
$class{'successful-dos'} = "medium";
$class{'successful-recon-largescale'} = "medium";
$class{'successful-recon-limited'} = "medium";
$class{'suspicious-filename-detect'} = "medium";
$class{'suspicious-login'} = "medium";
$class{'system-call-detect'} = "medium";
$class{'unusual-client-port-connection'} = "medium";
$class{'web-application-activity'} = "medium";
$class{'attempted-admin'} = "high";
$class{'attempted-user'} = "high";
$class{'shellcode-detect'} = "high";
$class{'successful-admin'} = "high";
$class{'successful-user'} = "high";
$class{'trojan-activity'} = "high";
$class{'unsuccessful-user'} = "high";
$class{'web-application-attack'} = "high";

# These values are hostility metric thresholds the Main alerts tab

my $HOSTILE =        1;
my $AGRESSIVE = 0.8 ;
my $CURIOUS =        0.5;
my $PRUDENT =        0.2;
my $SUSPECT =        0.1;

# This is the global attack probability (don't touch until you know what you're doing)

my $global_prob = 0.5 ;
```

```
##########################################################################################
# form variables handling
my $CGI=CGI->new();
my $USER=$CGI->param('user');


my $PASS=$CGI->param('pw');
my $HOST=$CGI->param('host');
my $DBNAME=$CGI->param('db');
my $LIMIT=$CGI->param('limit');
my $MINDATE=$CGI->param('mindate');
my $MAXDATE=$CGI->param('maxdate');
my $MEMPW=$CGI->param('mempw');
my $TARGET_DEL=$CGI->param('target_del');
my $TRUEIP=$CGI->param('trueip');
my $FALSEIP=$CGI->param('falseip');

if ($LIMIT eq ""){
                $LIMIT = "100";
}
if ($MINDATE eq ""){
                $MINDATE = date_only(time());
}

if ($MAXDATE eq ""){
                $MAXDATE = date_only(time());
}

my (        $HUMAN_DATE,
        $NUMBERHOST,@hosts,%hostility,%score,%bayes,%dispertion,%high,%medium,%low);
##########################################################################################

# Main program
$HUMAN_DATE=sapiens(time());
&head();
&form();
if ($PASS ne ""){

# comment the &activity_graph(); line if you don't want bargraph                    #


if ($TRUEIP) {
                &learn_true($TRUEIP);
                $TRUEIP = "";
        }
        if ($FALSEIP) {
                &learn_false($FALSEIP);
                $FALSEIP = "";
        }
        &refresh();
        &menu();
        &get_bayes();
        &statistics();
        &activity_graph();
        &attacks();
        &dispertion_graph();
        &sensors();
}else{
        &initialquery();
}

&foot();
###########################################################################################
# Loading events into memory and process hostility and dispertion criteria
sub statistics{
   my ($HOSTS_QUERY,
        $SIGNATURE_QUERY,
        $TIME_QUERY,
        $DBH,
        $STH,
        $NUMBERFIELDS,
        $REF,

        $a1,
        $b1,
        $weight_buffer,
        $low_buffer,
        $medium_buffer,
        $high_buffer,
        $i);

        $HOSTS_QUERY="
        select inet_ntoa(iphdr.ip_src), count(iphdr.cid)
        from iphdr, event
        where iphdr.cid=event.cid and left(event.timestamp, 10) >= \"$MINDATE\" and
left(event.timestamp, 10) <= \"$MAXDATE\"

        group by iphdr.ip_src;";

        $DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");
        $STH = $DBH->prepare($HOSTS_QUERY); # sending query to mysql
        $STH->execute();
        $NUMBERHOST = $STH->rows;
        $NUMBERFIELDS = $STH->{'NUM_OF_FIELDS'};        ## number of fields in the result
table
        # printing the result TAB
        while ($REF = $STH->fetchrow_arrayref) {
                push(@hosts,([$$REF[0],$$REF[1]]));
        }
        @hosts = sort {$a1 = $$a[1]; $b1 = $$b[1]; $b1 <=> $a1} @hosts;
        # now procceding with hostility
        for ($i = 0;  $i < $LIMIT;  $i++) {

                $SIGNATURE_QUERY="
                        select sig_class.sig_class_name,signature.sig_sid
                        from iphdr,event,signature,sig_class
```

```
                    where inet_ntoa(iphdr.ip_src)=\"$hosts[$i][0]\" and iphdr.cid=event.cid and
event.signature=signature.sig_id and sig_class.sig_class_id=signature.sig_class_id and
left(event.timestamp, 10) >= \"$MINDATE\" and left(event.timestamp, 10) <= \"$MAXDATE\"
                order by sig_class.sig_class_name;";

                $STH = $DBH->prepare($SIGNATURE_QUERY); # sending query to mysql
                $STH->execute();

                $weight_buffer = 0;
                $low_buffer = 0;
                $medium_buffer = 0;
                $high_buffer = 0;

                while ($REF = $STH->fetchrow_arrayref) {
                        if ($bayes{$$REF[1]}){
                                $weight_buffer += ($weight{$$REF[0]} * $bayes{$$REF[1]});
                        }else{
                                $weight_buffer += $weight{$$REF[0]};
                        }

                        if ($class{$$REF[0]} eq "low") {$low_buffer ++};
                        if ($class{$$REF[0]} eq "medium") {$medium_buffer ++};
                        if ($class{$$REF[0]} eq "high") {$high_buffer ++};
                }

                $hostility{"$hosts[$i][0]"} = $weight_buffer;
                $low{"$hosts[$i][0]"} = $low_buffer
                $medium{"$hosts[$i][0]"} = $medium_buffer;
                $high{"$hosts[$i][0]"} = $high_buffer;
                #print "$hosts[$i][0] --- $medium_buffer <BR>\n";
        }
        # now procceding with dispertion
        for ($i = 0;  $i < $LIMIT;  $i++) {


                $TIME_QUERY="
                select STD(UNIX_TIMESTAMP(event.timestamp)),
MIN(UNIX_TIMESTAMP(event.timestamp)), MAX(UNIX_TIMESTAMP(event.timestamp))
                from iphdr,event,signature,sig_class
                where inet_ntoa(iphdr.ip_src)=\"$hosts[$i][0]\" and iphdr.cid=event.cid and
left(event.timestamp, 10) >= \"$MINDATE\" and left(event.timestamp, 10) <= \"$MAXDATE\";";
                $STH = $DBH->prepare($TIME_QUERY); # sending query to mysql
                $STH->execute();

                $weight_buffer = 0;
                $low_buffer = 0;
                $medium_buffer = 0;
                $high_buffer = 0;

                while ($REF = $STH->fetchrow_arrayref) {
                        $dispertion{"$hosts[$i][0]"} = $$REF[0]/($$REF[2]-$$REF[1]+1);
                }
        }

        $STH->finish();
        $DBH->disconnect();
}

###########################################################################################
# Get bayes probability

sub get_bayes {
                my($FILE)="signature_bayes";
                my $line;
                my @buffer;
        open(PTR_FILE,$FILE) or die "Problem opening $FILE !";

        while($line=<PTR_FILE>) {


                @buffer = split (/:/,$line);

                # Bayes equation ##################
                if (($buffer[1])||($buffer[2])) {
                        $bayes{$buffer[0]} = 2*(($buffer[1] * $global_prob) / ( ($buffer[1]*
$global_prob) + ($buffer[2] * (1-$global_prob))));
                }
                #print "$buffer[0] $buffer[1] $buffer[2] $bayes{$buffer[0]} \n";
        }
        close(PTR_FILE);
}
###########################################################################################
# This one update the bayesian database from a true attack

sub learn_true{
  my ($SIGNATURE_QUERY,
        $DBH,
        $STH,
        $REF,
        $signature,
        $line,
        @buffer,
        %true_attack,
        %false_attack);

        my $IP = $_[0];
        my($FILE)="signature_bayes";

        open(PTR_FILE,$FILE) or die "Problem opening $FILE !";

        while($line=<PTR_FILE>) {
                chop($line);
                #if ($line =~ /^[0-9]+:[0-9]+:[0-9]+$/) {
```

```
				@buffer = split (/:/,$line);

				$true_attack{$buffer[0]} = $buffer[1];
				$false_attack{$buffer[0]} = $buffer[2];
			#}
	}
	close(PTR_FILE);

	$SIGNATURE_QUERY="
			select signature.sig_sid
			from iphdr,event,signature
			where inet_ntoa(iphdr.ip_src)=\"$IP\" and iphdr.cid=event.cid and
event.signature=signature.sig_id and left(event.timestamp, 10) >= \"$MINDATE\" and
left(event.timestamp, 10) <= \"$MAXDATE\"
			order by event.signature;";
	$DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");
	$STH = $DBH->prepare($SIGNATURE_QUERY); # sending query to mysql
	$STH->execute();

	while ($REF = $STH->fetchrow_arrayref) {
			if ($true_attack{$$REF[0]}){
				$true_attack{$$REF[0]} += 0.01;
			}else{
				$true_attack{$$REF[0]} = 0.5;
				$true_attack{$$REF[0]} += 0.01;
				$false_attack{$$REF[0]} = 0.5;
			}
	}
	open(PTR_FILE, ">$FILE") or die "Problem opening $FILE !";
	foreach $signature (keys(%true_attack)) {
			#print "$signature\n";
			print PTR_FILE
"$signature".':'."$true_attack{$signature}".':'."$false_attack{$signature}".':'."\n";
	}

	close(PTR_FILE);
	$STH->finish();
	$DBH->disconnect();
}
##########################################################################################
# This one update the bayesian database from a false positive

sub learn_false{
  my ($SIGNATURE_QUERY,
	$DBH,
	$STH,
	$REF,
	$signature,
	$line,
	@buffer,
	%true_attack,
	%false_attack);
```

```
        my $IP = $_[0];
        my($FILE)="signature_bayes";

        open(PTR_FILE,$FILE) or die "Problem opening $FILE !";

        while($line=<PTR_FILE>) {
                chop($line);
                #if ($line =~ /^[0-9]+:[0-9]+:[0-9]+$/) {

                        @buffer = split (/:/,$line);

                        $true_attack{$buffer[0]} = $buffer[1];
                        $false_attack{$buffer[0]} = $buffer[2];
                #}
        }
        close(PTR_FILE);

        $SIGNATURE_QUERY="
                select signature.sig_sid
                from iphdr,event,signature
                where inet_ntoa(iphdr.ip_src)=\"$IP\" and iphdr.cid=event.cid and
event.signature=signature.sig_id and left(event.timestamp, 10) >= \"$MINDATE\" and
left(event.timestamp, 10) <= \"$MAXDATE\"
                order by event.signature;";

        $DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");
        $STH = $DBH->prepare($SIGNATURE_QUERY); # sending query to mysql
        $STH->execute();

        while ($REF = $STH->fetchrow_arrayref) {
                if ($false_attack{$$REF[0]}){
                        $false_attack{$$REF[0]} += 0.01;
                }else{
                        $false_attack{$$REF[0]} = 0.5;
                        $false_attack{$$REF[0]} += 0.01;
                        $true_attack{$$REF[0]} = 0.5;
                }
        }

        open(PTR_FILE, ">$FILE") or die "Problem opening $FILE !";

        foreach $signature (keys(%false_attack)) {
                print PTR_FILE
"$signature".':'."$true_attack{$signature}".':'."$false_attack{$signature}".':'."\n";
        }
        close(PTR_FILE);
        $STH->finish();
        $DBH->disconnect();
}
##########################################################################################
```

```
# Sources IP sorted / number of alerts
sub attacks{
        my ($i,
                @COLUMN,
                $SIP,
                $color);

        @COLUMN=("Whois query","World map","SOURCE
IP\@","Low","Medium","High","Hostility", "Bayes true", "Bayes false");
        # printing the result TAB

        print "<a name=\"0\"></a>";
        print "<H3>Attack source(s) from $MINDATE to $MAXDATE :</H3>\n";
        print "
                <CENTER>\n
                <TABLE BORDER=0 CELLPADDING=6 bgcolor=\"#aeaaae\" width=\"100\%\">\n
                <TR bgcolor=\"#00008b\">";

        for ($i = 0;  $i < 9;  $i++) {
                        print "<TD ALIGN=center><B><FONT size=\"1\"
color=\"#ffffff\">@COLUMN[$i]</FONT></B></TD>";
        }

        print "</TR>\n";
                for ($i = 0;  $i < $LIMIT;  $i++) {
                        $SIP = $hosts[$i][0];
                        CASE: {
                          ($hostility{$SIP} >= $HOSTILE) && do{
                                  $color="#bf0000";
                                  last CASE;
                          };
                          ($hostility{$SIP} >= $AGRESSIVE) && do{
                                  $color="#ff7f00";


                                  last CASE;
                          };
                          ($hostility{$SIP} >= $CURIOUS) && do{
                                  $color="#fff600";
                                  last CASE;
                          };

                          ($hostility{$SIP} >= $PRUDENT) && do{
                                  $color="#0000bf";
                                  last CASE;
                          };

                          ($hostility{$SIP} >= $SUSPECT) && do{
                                  $color="#5ba8f5";
                                  last CASE;
                          };

                          ($hostility{$SIP} < $SUSPECT) && do{
```

```
                                        $color="#75C4F5";
                                        last CASE;
                                };
                        } # end of CASE block

                                if ($i % 2) {
                                                        print "<TR bgcolor=\"#cccccc\">";
                                                }else{
                                                        print "<TR bgcolor=\"#bbbbbb\">";
                                                }
                                print "
                                <TD ALIGN=center><A
HREF=\"http://www.ripe.net/perl/whois\?searchtext=".$SIP."\" TARGET=new>
                                <IMG BORDER=0 width=24 height=24 SRC=\"./ripe.gif\"
ALT=\"Whois\"></A></TD>


                                <TD ALIGN=center><A HREF=\"http://www.antionline.com/tools-and-toys/ip-
locate/index.php?address=".$SIP."\" TARGET=new>
                                <IMG BORDER=0 width=24 height=24 SRC=\"./world.gif\"
ALT=\"Whois\"></A></TD>

                                <TD ALIGN=center><FONT size=\"1\">
                                <FORM ACTION=\"alerts.pl\" METHOD= post > \n
                                <INPUT TYPE=\"hidden\" NAME=\"pw\" VALUE=".$PASS.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"mempw\" VALUE=".$MEMPW.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"host\" VALUE=".$HOST.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"db\" VALUE=".$DBNAME.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"user\" VALUE=".$USER.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"hacker\" VALUE=".$SIP.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"limit\" VALUE=".$LIMIT.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"mindate\" VALUE=".$MINDATE.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"maxdate\" VALUE=".$MAXDATE.">\n
                                <INPUT TYPE=\"submit\" NAME=\"submitButtonName\" VALUE=$SIP>\n
                                </FORM></FONT></TD>\n

                                <TD ALIGN=center><FONT size=\"1\">".$low{$SIP}."</FONT></TD>\n
                                <TD ALIGN=center><FONT size=\"1\">".$medium{$SIP}."</FONT></TD>\n
                                <TD ALIGN=center><FONT size=\"1\">".$high{$SIP}."</FONT></TD>\n
                                <TD ALIGN=center bgcolor=\"$color\"><FONT size=\"1\"> $hostility{$SIP}
</FONT></TD>

                                <TD ALIGN=center><FONT size=\"1\">
                                <FORM ACTION=\"snort.pl\" METHOD= post > \n
                                <INPUT TYPE=\"hidden\" NAME=\"pw\" VALUE=".$PASS.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"mempw\" VALUE=".$MEMPW.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"host\" VALUE=".$HOST.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"db\" VALUE=".$DBNAME.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"user\" VALUE=".$USER.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"hacker\" VALUE=".$SIP.">\n
                                <INPUT TYPE=\"hidden\" NAME=\"limit\" VALUE=".$LIMIT.">\n
```

```
                    <INPUT TYPE=\"hidden\" NAME=\"mindate\" VALUE=".$MINDATE.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"maxdate\" VALUE=".$MAXDATE.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"trueip\" VALUE=".$SIP.">\n
                    <INPUT TYPE=\"submit\" NAME=\"submitButtonName\" VALUE=\"true\">\n

                    </FORM></FONT></TD>
                    <TD ALIGN=center><FONT size=\"1\">

                    <FORM ACTION=\"snort.pl\" METHOD= post > \n

                    <INPUT TYPE=\"hidden\" NAME=\"pw\" VALUE=".$PASS.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"mempw\" VALUE=".$MEMPW.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"host\" VALUE=".$HOST.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"db\" VALUE=".$DBNAME.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"user\" VALUE=".$USER.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"hacker\" VALUE=".$SIP.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"limit\" VALUE=".$LIMIT.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"mindate\" VALUE=".$MINDATE.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"maxdate\" VALUE=".$MAXDATE.">\n
                    <INPUT TYPE=\"hidden\" NAME=\"falseip\" VALUE=".$SIP.">\n
                    <INPUT TYPE=\"submit\" NAME=\"submitButtonName\"
VALUE=\"false\">\n

                    </FORM></FONT></TD>\n";
                    print "</TR>\n";
            }

        print "
        </TABLE></CENTER>\n";
}
####################################################################################



# Activity graph  (10 most active IP)

sub activity_graph{
        my ($i, @LOW_ALERTS, @MEDIUM_ALERTS, @HIGH_ALERTS, @IP, $NUM_ANGRY,
        $DBH, $STH, $REF, @data, $GRAPH, $IMAGE, $png);

        for ($i = 0;  $i < 10;  $i++) {
                if ($hosts[$i][1] ne ""){
                        @IP[$i] = $hosts[$i][0];
                        @LOW_ALERTS[$i] = $low{"$hosts[$i][0]"};
                        @MEDIUM_ALERTS[$i] = $medium{"$hosts[$i][0]"};
                        @HIGH_ALERTS[$i] = $high{"$hosts[$i][0]"};
                        $NUM_ANGRY=$i+1;
                }else{
                        @IP[$i] ="";
                        @LOW_ALERTS[$i]="";
                        @MEDIUM_ALERTS[$i]="";
                        @HIGH_ALERTS[$i]="";
                }
        }

        @data = ([@IP], [@LOW_ALERTS], [@MEDIUM_ALERTS], [@HIGH_ALERTS]);

        $GRAPH = GD::Graph::bars->new(850, 200);
        $GRAPH->set( dclrs => [ qw(blue orange red) ] );
        $GRAPH->set(

        x_label     => 'Hosts',
        y_label     => 'Number of alerts by type',
        title       => 'Activity',

        bar_width  => 5,
        show_values => 1,


        bar_spacing    => 8,
        shadow_depth   => 2,

        shadowclr      => 'grey75',

        ) or warn $GRAPH->error;

        $GRAPH->set_legend_font(GD::gdMediumBoldFont);
        $GRAPH->set_x_axis_font(GD::gdTinyFont);
        $GRAPH->set_y_axis_font(GD::gdMediumBoldFont);
        $GRAPH->set_x_label_font(GD::gdMediumBoldFont);
        $GRAPH->set_y_label_font(GD::gdMediumBoldFont);
        $GRAPH->set_title_font(GD::gdMediumBoldFont);
        $GRAPH->set_legend( 'Low', 'Medium', 'High' );
        $IMAGE = $GRAPH->plot(\@data) or die $GRAPH->error;

        $png=$IMAGE->png;
        open (IMAGE,">img1.png");
        binmode IMAGE;
        print IMAGE $png;
        close(IMAGE);

        print "<H3>$NUM_ANGRY most active host(s) from $MINDATE to $MAXDATE :</H3>\n";
        print "<a name=\"1\"></a>";
        print "<CENTER><TABLE BORDER=0 CELLPADDING=6
bgcolor=\"#ffffff\"><TR><TD>\n";
        print "<CENTER><IMG BORDER=1 SRC=\"img1.png\"></IMG></CENTER>\n";
        print "</TD></TR></TABLE></CENTER>\n";
}
#######################################################################################
# Dispertion graph

sub dispertion_graph{

        my ($i, @DISP, @IP, $NUM_ANGRY, $DBH, $STH, $REF, @data, $GRAPH, $IMAGE,
                $png);
```

```
	for ($i = 0;  $i < 10;  $i++) {
		if ($hosts[$i][1] ne ""){
			@IP[$i] = $hosts[$i][0];
			@DISP[$i] = $dispertion{"$hosts[$i][0]"};
			$NUM_ANGRY=$i+1;
		}else{
			@IP[$i] ="";
			@DISP[$i]="";
		}
	}
	@data = ([@IP], [@DISP]);
	$GRAPH = GD::Graph::hbars->new(850, 200);
	$GRAPH->set( dclrs => [ qw(grey blue) ] );
	$GRAPH->set(

	x_label      => 'Hosts',
	y_label      => 'Dispertion',
	title        => 'Dispertion of alerts',
	#bar_spacing => 4,
	#bar_width   => 5,
	show_values => 1
	) or warn $GRAPH->error;

	$GRAPH->set_legend_font(GD::gdMediumBoldFont);
	$GRAPH->set_x_axis_font(GD::gdTinyFont);
	$GRAPH->set_y_axis_font(GD::gdMediumBoldFont);
	$GRAPH->set_x_label_font(GD::gdMediumBoldFont);
	$GRAPH->set_y_label_font(GD::gdMediumBoldFont);
	$GRAPH->set_title_font(GD::gdMediumBoldFont);
	$GRAPH->set_legend( 'Standard deviation (sec)' );

	$IMAGE = $GRAPH->plot(\@data) or die $GRAPH->error;

	$png=$IMAGE->png;
	open (IMAGE,">img2.png");

	binmode IMAGE;
	print IMAGE $png;
	close(IMAGE);

	print "<H3>Time dispertion of alerts from $MINDATE to $MAXDATE :</H3>\n";
	print "<a name=\"2\"></a>";
	print "<CENTER><TABLE BORDER=0 CELLPADDING=6
bgcolor=\"#ffffff\"><TR><TD>\n";
	print "<CENTER><IMG BORDER=1 SRC=\"img2.png\"></IMG></CENTER>\n";
	print "</TD></TR></TABLE></CENTER>\n";
}
###########################################################################################
# Sensors report
sub sensors{
	my ($SENSORS_QUERY, $i, $DBH, $STH, $NUMBERFIELDS, $NUMBEROW, @COLUMN,
	$REF);
	$SENSORS_QUERY="
		SELECT sid,hostname,interface
		FROM sensor
		ORDER BY sid;";

	$DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");
	$STH = $DBH->prepare($SENSORS_QUERY); # sending query to mysql
	$STH->execute();

	$NUMBEROW = $STH->rows;
	$NUMBERFIELDS = $STH->{'NUM_OF_FIELDS'};		## number of fields in the result
table
	# $COLUMNAME = $STH->{'NAME'};


@COLUMN=("SENSOR ID","ADDRESS","INTERFACE");
	# printing the result TAB
	print "<a name=\"3\"></a>";
	print "<H3>$NUMBEROW Sensor(s) :</H3>\n";
	print " <CENTER>\n
	<TABLE BORDER=0 CELLPADDING=6 bgcolor=\"#00008b\" width=\"100\%\">\n
	<TR>";

	for ($i = 0;  $i < 3;  $i++) {
	    print "<TD ALIGN=center><B><FONT size=\"1\"
color=\"#ffffff\">@COLUMN[$i]</FONT></B></TD></FONT>";
	}
	print "</TR>\n";
	while ($REF = $STH->fetchrow_arrayref) {
		print "<TR>";
		for ($i = 0;  $i < 1;  $i++) {
			print "<TD ALIGN=center><FONT size=\"1\"
color=\"#ffffff\">$$REF[$i]</FONT></TD>";
			}
		for ($i = 1;  $i < 2;  $i++) {
			print "<TD ALIGN=center><FONT size=\"1\"
color=\"#ffffff\">$$REF[$i]</FONT></TD>";
		}
		for ($i = 2;  $i < 3;  $i++) {
			print "<TD ALIGN=center><FONT size=\"1\"
color=\"#ffffff\">$$REF[$i]</FONT></TD>";
		}
		print "</TR>\n";
	}
	print " </TABLE></CENTER>\n";
	#print "<P><B>sql :  </B>".$SENSORS_QUERY."</p>\n";  # debug
	$STH->finish();
	$DBH->disconnect();
}

# FORM
sub form {
	print "
```

```
        <H3>General parameters :</H3> \n
        <FORM ACTION=\"snort.pl\" METHOD= post > \n
        <CENTER><TABLE BORDER=0 CELLPADDING=5 bgcolor=\"#00008b\"
width=\"100\%\"><TR> \n";
        if ($HOST ne ""){
                print "<TD align=center ><FONT color=\"#ffffff\"><b>DB Host
:</b></FONT><INPUT TYPE=\"text\" NAME=\"host\" VALUE=".$HOST." size=\"24\"> </TD>";
        }else {
                print "<TD align=center ><FONT color=\"#ffffff\"><b>DB Host
:</b></FONT><INPUT TYPE=\"text\" NAME=\"host\" size=\"24\"> </TD>";
        }
        if ($DBNAME ne ""){
                print "<TD align=center ><FONT color=\"#ffffff\"><b>DB Name
:</b></FONT><INPUT TYPE=\"text\" NAME=\"db\" VALUE=".$DBNAME." size=\"24\"> </TD>";
        }else {
                print "<TD align=center ><FONT color=\"#ffffff\"><b>DB Name
:</b></FONT><INPUT TYPE=\"text\" NAME=\"db\" size=\"24\"> </TD>";
        }
        if ($USER ne ""){
                print "<TD align=center ><FONT color=\"#ffffff\"><b>DB User
:</b></FONT><INPUT TYPE=\"text\" NAME=\"user\" VALUE=".$USER." size=\"24\"> </TD>";
        }else{
                print "<TD align=center ><FONT color=\"#ffffff\"><b>DB User
:</b></FONT><INPUT TYPE=\"text\" NAME=\"user\" size=\"24\"> </TD>";
        }
        if ($PASS ne "" && $MEMPW eq "on") {
                print "<TD align=center ><FONT color=\"#ffffff\"><b>DB Pass :</b></FONT><INPUT
TYPE=\"password\" NAME=\"pw\" VALUE=".$PASS." size=\"24\"> </TD>";
        }else{
                print "<TD align=center ><FONT color=\"#ffffff\"><b>DB Pass :</b></FONT><INPUT
TYPE=\"password\" NAME=\"pw\" size=\"24\"> </TD>";

        }
        if ($MEMPW ne "on") {
                print "<TD align=center ><FONT color=\"#ffffff\"><b>Remember PW
?</b></FONT><INPUT TYPE=\"checkbox\" NAME=mempw > </TD>";
        }else{
                print "<TD align=center ><FONT color=\"#ffffff\"><b>Remember PW
?</b></FONT><INPUT TYPE=\"checkbox\" NAME=mempw CHECKED > </TD>";
        }
        print "
        </TR></TABLE> </CENTER>\n";
}

sub menu {
        print "
        <TABLE BORDER=0 CELLPADDING=6 bgcolor=\"#bbbbbb\" width=\"100\%\" ><TR> \n
        <TD ALIGN=center><B><FONT color=\"#339966\"><a color=\"#000000\" href=\"#1\">Activity
graph</a></FONT></B></TD>
        <TD ALIGN=center><B><FONT color=\"#339966\"><a color=\"#000000\" href=\"#0\">Sources
classification</a></FONT></B></TD>
        <TD ALIGN=center><B><FONT color=\"#339966\"><a color=\"#000000\"
href=\"#2\">Dispertion graph</a></FONT></B></TD>
```

```
        <TD ALIGN=center><B><FONT color=\"#339966\"><a color=\"#000000\"
href=\"#3\">Sensors</a></FONT></B></TD>
        </TD></TR></TABLE><BR> \n";
}

sub initialquery {
        print "
        <TABLE BORDER=0 CELLPADDING=6 bgcolor=\"#00008b\" width=\"100\%\" ><TR> \n
        <TD ALIGN=center rowspan=2> \n
        <P><INPUT TYPE=\"submit\" NAME=\"submitButtonName\" VALUE=\"Send query\"> \n
 </P> \n
        </TD></TR>\n


</TABLE></FORM> \n
   <BR> \n";
}
sub refresh {
        print "
        <TABLE BORDER=0 CELLPADDING=6 bgcolor=\"#00008b\" width=\"100\%\" ><TR> \n
        <TD align=center ><FONT color=\"#ffffff\"><b>Min date : </b></FONT><INPUT
TYPE=\"text\" name=\"mindate\" value=$MINDATE size=\"10\"> </TD>
        <TD align=center ><FONT color=\"#ffffff\"><b>Max date : </b></FONT><INPUT
TYPE=\"text\" name=\"maxdate\" value=$MAXDATE size=\"10\"> </TD>
        <TD ALIGN=center> \n
        <TD align=center ><FONT color=\"#ffffff\"><b>Limit : </b></FONT><INPUT TYPE=\"text\"
name=\"limit\" value=$LIMIT size=\"10\"> </TD>
        <TD ALIGN=center> \n
        <P><INPUT TYPE=\"submit\" NAME=\"submitButtonName\" VALUE=\"Refresh\"> \n
        </TD></TR></TABLE></FORM><BR> \n";
}
###############################################################################
# Header
sub head{
        print "
        <HTML><HEAD>\n
        <BODY lang=FR bgcolor=\"#f4f4e0\" style='FONT-size:08.0pt\;FONT-family:Sans'> \n
        <TABLE ALIGN=center BORDER=0 CELLPADDING=6 bgcolor=\"#f4f4e0\" width=\"100\%\"
><TR> \n
        <TD><CENTER><A HREF=\"http://www.ripe.net/\" TARGET=new>
        <IMG BORDER=0 width=50 height=22 SRC=\"./ripe.gif\"
ALT=\"Whois\"></A></CENTER></TD> \n
        <TD><CENTER><A HREF=\"http://www.nic.fr/zonecheck/\" TARGET=new>
        <IMG BORDER=0 width=50 height=22 SRC=\"./afnic.gif\"
ALT=\"Zonecheck\"></A></CENTER></TD> \
        <TD><CENTER><A HREF=\"http://www.snort.org/snort-db/\" TARGET=new>
        <IMG BORDER=0 width=50 height=22 SRC=\"./snort.gif\"
ALT=\"SnortDB\"></A></CENTER></TD> \n

        <TD><CENTER><A HREF=\"http://online.securityfocus.com/bid/\" TARGET=new>
        <IMG BORDER=0 width=50 height=22 SRC=\"./focus.gif\"
ALT=\"SecurityFocus\"></A></CENTER></TD> \n
        </TR></TABLE> \n
        <TITLE>SNORT</TITLE>\n
        <A name=top></A> \n
        <H1 ALIGN=center>SNORT Reporter</H1> \n
        </HEAD> \n
        <P ALIGN=center>Reporting and investigation for the SNORT Network Intrusion Detection
System.</P> \n";
}
###############################################################################################
# Footer
sub foot{
        print "
        Report created on : $HUMAN_DATE </P> \n
        <P ALIGN=center><A HREF=\"\#top\">Top of the page</a></P> \n
        </BODY></HTML>";
}
###############################################################################################
# date formating
sub sapiens {
        my
($HUMAN_DATE,$SEC,$MIN,$HOUR,$MDAY,$MONTH,$YEAR,$SDAY,$ADAY,$ISDST);
        ($SEC,$MIN,$HOUR,$MDAY,$MONTH,$YEAR,$SDAY,$ADAY,$ISDST) = localtime($_[0]);
        $YEAR = ($YEAR-100);
        $YEAR ="200$YEAR";
        $MONTH++;
        if ($MONTH < 10){
                $MONTH="0".$MONTH;
        }
        if ($MDAY < 10){
                $MDAY="0".$MDAY;
        }

        if ($HOUR < 10){
                $HOUR="0".$HOUR;
        }

        if ($MIN < 10){
                $MIN="0".$MIN;
        }
        if ($SEC < 10){
                $SEC="0".$SEC;
        }
        #@LIST = ('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec');
        #$MONTH = $LIST[$MONTH];
        $HUMAN_DATE = "$YEAR-$MONTH-$MDAY $HOUR:$MIN:$SEC ";
return $HUMAN_DATE
}

sub date_only {
        my ($DATE,$SEC,$MIN,$HOUR,$MDAY,$MONTH,$YEAR,$SDAY,$ADAY,$ISDST);
        ($SEC,$MIN,$HOUR,$MDAY,$MONTH,$YEAR,$SDAY,$ADAY,$ISDST) = localtime($_[0]);
        $YEAR = ($YEAR-100);
        $YEAR ="200$YEAR";
        $MONTH++;
        if ($MONTH < 10){
                $MONTH="0".$MONTH;
        }
        if ($MDAY < 10){
                $MDAY="0".$MDAY;
        }
        if ($HOUR < 10){
                $HOUR="0".$HOUR;
        }
        if ($MIN < 10){
                $MIN="0".$MIN;
        }

        if ($SEC < 10){
                $SEC="0".$SEC;
        }
        #@LIST = ('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec');
        #$MONTH = $LIST[$MONTH];
        $DATE = "$YEAR-$MONTH-$MDAY";
return $DATE
}
```

Showing Signature Bayes

```
1:0.5:0.5
2:0.5:0.5
3:0.5:0.5
4:0.5:0.5
5:0.5:0.5
6:0.5:0.5
7:0.5:0.5
8:0.5:0.5
9:0.5:0.5
10:0.5:0.5
11:0.5:0.5
12:0.5:0.5
13:0.5:0.5
14:0.5:0.5
15:0.5:0.5
16:0.5:0.5
17:0.5:0.5
18:0.5:0.5
19:0.5:0.5
20:0.5:0.5
21:0.5:0.5
22:0.5:0.5
23:0.5:0.5
24:0.5:0.5
25:0.5:0.5
```

```
2572:0.5:0.5
2573:0.5:0.5
2574:0.5:0.5
2575:0.5:0.5
2576:0.5:0.5
2577:0.5:0.5
2578:0.5:0.5
2579:0.5:0.5
2580:0.5:0.5
2581:0.5:0.5
2582:0.5:0.5
2583:0.5:0.5
2584:0.5:0.5
2585:0.5:0.5
2586:0.5:0.5
2587:0.5:0.5
2588:0.5:0.5
2589:0.5:0.5
2590:0.5:0.5
2591:0.5:0.5
2592:0.5:0.5
2593:0.5:0.5
2594:0.5:0.5
2595:0.5:0.5
2596:0.5:0.5
2597:0.5:0.5
2598:0.5:0.5
2599:0.5:0.5
2600:0.5:0.5
```

Showing Miniserver.pl

```
#!/usr/bin/perl
# A very simple perl web server used by Webmin

# Require basic libraries
package miniserv;
use Socket;
use POSIX;

# Find and read config file
if (@ARGV != 1) {
        die "Usage: miniserv.pl <config file>";
        }
if ($ARGV[0] =~ /^\//) {
        $conf = $ARGV[0];
        }
else {
        chop($pwd = `pwd`);
```

```
		$conf = "$pwd/$ARGV[0]";
		}
open(CONF, $conf) || die "Failed to open config file $conf : $!";
while(<CONF>) {
		s/\r|\n//g;
		if (/^#/ || !/\S/) { next; }

		/^([^=]+)=(.*)$/;
		$name = $1; $val = $2;
		$name =~ s/^\s+//g; $name =~ s/\s+$//g;
		$val =~ s/^\s+//g; $val =~ s/\s+$//g;
		$config{$name} = $val;
		}
close(CONF);

# Check is SSL is enabled and available
if ($config{'ssl'}) {
		eval "use Net::SSLeay";
		if (!$@) {
				$use_ssl = 1;
				# These functions only exist for SSLeay 1.0
				eval "Net::SSLeay::SSLeay_add_ssl_algorithms()";
				eval "Net::SSLeay::load_error_strings()";
				if (defined(&Net::SSLeay::X509_STORE_CTX_get_current_cert) &&
				   defined(&Net::SSLeay::CTX_load_verify_locations) &&
				   defined(&Net::SSLeay::CTX_set_verify)) {
						$client_certs = 1;
						}
				}
		}

# Check if the syslog module is available to log hacking attempts
if ($config{'syslog'} && !$config{'inetd'}) {
		eval "use Sys::Syslog qw(:DEFAULT setlogsock)";
		if (!$@) {
				$use_syslog = 1;
				}
		}

# check if the TCP-wrappers module is available
if ($config{'libwrap'}) {
		eval "use Authen::Libwrap qw(hosts_ctl STRING_UNKNOWN)";
		if (!$@) {
				$use_libwrap = 1;
				}
		}

# Get miniserv's perl path and location
$miniserv_path = $0;
open(SOURCE, $miniserv_path);
<SOURCE> =~ /^#!(\S+)/; $perl_path = $1;
close(SOURCE);
@miniserv_argv = @ARGV;
```

```
# Check vital config options
%vital = ("port", 80,
            "root", "./",
            "server", "MiniServ/0.01",
            "index_docs", "index.html index.htm index.cgi index.pl index.php",
            "addtype_html", "text/html",
            "addtype_txt", "text/plain",
            "addtype_gif", "image/gif",
            "addtype_jpg", "image/jpeg",
            "addtype_jpeg", "image/jpeg",
            "realm", "MiniServ",
            "session_login", "/session_login.cgi",
            "password_form", "/password_form.cgi",
            "password_change", "/password_change.cgi",
            "maxconns", 50,
            "pam", "webmin",
            "sidname", "sid",
            "unauth", "^/unauthenticated/ ^[A-Za-z0-9\\-/]+\\.jar\$ ^[A-Za-z0-9\\-/]+\\.class\$ ^[A-Za-z0-9\\-
/]+\\.gif\$ ^[A-Za-z0-9\\-/]+\\.conf\$",
            "max_post", 10000
           );
foreach $v (keys %vital) {
           if (!$config{$v}) {
                      if ($vital{$v} eq "") {
                                 die "Missing config option $v";
                                 }
                      $config{$v} = $vital{$v};
                      }
           }
if (!$config{'sessiondb'}) {
           $config{'pidfile'} =~ /^(.*)\/[^\/]+$/;
           $config{'sessiondb'} = "$1/sessiondb";
           }
if (!$config{'errorlog'}) {
           $config{'logfile'} =~ /^(.*)\/[^\/]+$/;
           $config{'errorlog'} = "$1/miniserv.error";
           }
$sidname = $config{'sidname'};
die "Session authentication cannot be used in inetd mode"
           if ($config{'inetd'} && $config{'session'});

# check if the PAM module is available to authenticate
if (!$config{'no_pam'}) {
           eval "use Authen::PAM";
           if (!$@) {
                      # check if the PAM authentication can be used by opening a
                      # PAM handle
                      local $pamh;
                      if (ref($pamh = new Authen::PAM($config{'pam'}, "root",
                                                                         \&pam_conv_func))) {
                                 # Now test a login to see if /etc/pam.d/XXX is set
                                 # up properly.
```

```
                        $pam_conv_func_called = 0;
                        $pam_username = "test";
                        $pam_password = "test";
                        $pamh->pam_authenticate();
                        if ($pam_conv_func_called) {
                                $pam_msg = "PAM authentication enabled";
                                $use_pam = 1;
                                }
                        else {
                                $pam_msg = "PAM test failed - maybe /etc/pam.d/$config{'pam'} does
not exist";
                                }
                        }
                else {
                        $pam_msg = "PAM initialization of Authen::PAM failed";
                        }
                }
        }

# init days and months for http_date
@weekday = ( "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" );
@month = ( "Jan", "Feb", "Mar", "Apr", "May", "Jun",
          "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" );

# Change dir to the server root
chdir($config{'root'});
$user_homedir = (getpwuid($<))[7];

# Read users file
if ($config{'userfile'}) {
        open(USERS, $config{'userfile'});
        while(<USERS>) {
                s/\r|\n//g;
                local @user = split(/:/, $_);
                $users{$user[0]} = $user[1];
                $certs{$user[0]} = $user[3] if ($user[3]);
                if ($user[4] =~ /^allow\s+(.*)/) {
                        $allow{$user[0]} = $config{'alwaysresolve'} ?
                                [ split(/\s+/, $1) ] :
                                [ &to_ipaddress(split(/\s+/, $1)) ];
                        }
                elsif ($user[4] =~ /^deny\s+(.*)/) {
                        $deny{$user[0]} = $config{'alwaysresolve'} ?
                                [ split(/\s+/, $1) ] :
                                [ &to_ipaddress(split(/\s+/, $1)) ];
                        }
                }
        close(USERS);
        }

# Setup SSL if possible and if requested
if (!-r $config{'keyfile'} ||
  $config{'certfile'} && !-r $config{'certfile'}) {
        # Key file doesn't exist!
        $use_ssl = 0;
        }
if ($use_ssl) {
        $ssl_ctx = Net::SSLeay::CTX_new() ||
                die "Failed to create SSL context : $!";
        $client_certs = 0 if (!-r $config{'ca'} || !%certs);
        if ($client_certs) {
                Net::SSLeay::CTX_load_verify_locations(
                        $ssl_ctx, $config{'ca'}, "");
                Net::SSLeay::CTX_set_verify(
                        $ssl_ctx, &Net::SSLeay::VERIFY_PEER, \&verify_client);
                }
        if ($config{'extracas'}) {
                foreach $p (split(/\s+/, $config{'extracas'})) {
                        Net::SSLeay::CTX_load_verify_locations(
                                $ssl_ctx, $p, "");
                        }
                }

        Net::SSLeay::CTX_use_RSAPrivateKey_file(
                $ssl_ctx, $config{'keyfile'},
                &Net::SSLeay::FILETYPE_PEM) || die "Failed to open SSL key";
        Net::SSLeay::CTX_use_certificate_file(
                $ssl_ctx, $config{'certfile'} || $config{'keyfile'},
                &Net::SSLeay::FILETYPE_PEM) || die "Failed to open SSL cert";
        }

# Setup syslog support if possible and if requested
if ($use_syslog) {
        eval 'openlog($config{"pam"}, "cons,pid,ndelay", "authpriv"); setlogsock("unix")';
        if ($@) {
                $use_syslog = 0;
                }
        else {
                local $msg = ucfirst($config{'pam'})." starting";
                eval { syslog("info", $msg); };
                if ($@) {
                        eval {
                                setlogsock("inet");
                                syslog("info", $msg);
                                };
                        if ($@) {
                                # All attempts to use syslog have failed..
                                $use_syslog = 0;
                                }
                        }
                }
        }

# Read MIME types file and add extra types
if ($config{"mimetypes"} ne "") {
        open(MIME, $config{"mimetypes"});
```

```
          while(<MIME>) {
                    chop; s/#.*$//;
                    if (/^(\S+)\s+(.*)$/) {
                              $type = $1; @exts = split(/\s+/, $2);
                              foreach $ext (@exts) {
                                        $mime{$ext} = $type;
                                        }
                              }
                    }
          close(MIME);
          }
foreach $k (keys %config) {
          if ($k !~ /^addtype_(.*)$/) { next; }
          $mime{$1} = $config{$k};
          }

# get the time zone
if ($config{'log'}) {
          local(@gmt, @lct, $days, $hours, $mins);
          @make_date_marr = ("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                               "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
          @gmt = gmtime(time());
          @lct = localtime(time());
          $days = $lct[3] - $gmt[3];
          $hours = ($days < -1 ? 24 : 1 < $days ? -24 : $days * 24) +
                    $lct[2] - $gmt[2];
          $mins = $hours * 60 + $lct[1] - $gmt[1];
          $timezone = ($mins < 0 ? "-" : "+"); $mins = abs($mins);
          $timezone .= sprintf "%2.2d%2.2d", $mins/60, $mins%60;
          }

# build anonymous access list
foreach $a (split(/\s+/, $config{'anonymous'})) {
          if ($a =~ /^([^=]+)=(\S+)$/) {
                    $anonymous{$1} = $2;
                    }
          }

# build unauthenticated URLs list
@unauth = split(/\s+/, $config{'unauth'});

# build redirect mapping
foreach $r (split(/\s+/, $config{'redirect'})) {
          if ($r =~ /^([^=]+)=(\S+)$/) {
                    $redirect{$1} = $2;
                    }
          }

# start up external authentication program, if needed
if ($config{'extauth'}) {
          socketpair(EXTAUTH, EXTAUTH2, AF_UNIX, SOCK_STREAM, PF_UNSPEC);
          if (!($extauth = fork())) {
                    close(EXTAUTH);
                    close(STDIN);
                    close(STDOUT);
                    open(STDIN, "<&EXTAUTH2");
                    open(STDOUT, ">&EXTAUTH2");
                    exec($config{'extauth'});
                    print STDERR "exec failed : $!\n";
                    exit 1;
                    }
          close(EXTAUTH2);
          local $os = select(EXTAUTH);
          $| = 1; select($os);
          }

# Re-direct STDERR to a log file
if ($config{'errorlog'} ne '-') {
          open(STDERR, ">>$config{'errorlog'}") || die "failed to open $config{'errorlog'} : $!";
          }

# Init allow and deny lists
@deny = split(/\s+/, $config{"deny"});
@deny = &to_ipaddress(@deny) if (!$config{'alwaysresolve'});
@allow = split(/\s+/, $config{"allow"});
@allow = &to_ipaddress(@allow) if (!$config{'alwaysresolve'});
if ($config{'allowusers'}) {
          @allowusers = split(/\s+/, $config{'allowusers'});
          }
elsif ($config{'denyusers'}) {
          @denyusers = split(/\s+/, $config{'denyusers'});
          }

if ($config{'inetd'}) {
          # We are being run from inetd - go direct to handling the request
          $SIG{'HUP'} = 'IGNORE';
          $SIG{'TERM'} = 'DEFAULT';
          $SIG{'PIPE'} = 'DEFAULT';
          open(SOCK, "+>&STDIN");

          # Check if it is time for the logfile to be cleared
          if ($config{'logclear'}) {
                    local $write_logtime = 0;
                    local @st = stat("$config{'logfile'}.time");
                    if (@st) {
                              if ($st[9]+$config{'logtime'}*60*60 < time()){
                                        # need to clear log
                                        $write_logtime = 1;
                                        unlink($config{'logfile'});
                                        }
                              }
                    else { $write_logtime = 1; }
                    if ($write_logtime) {
                              open(LOGTIME, ">$config{'logfile'}.time");
                              print LOGTIME time(),"\n";
                              close(LOGTIME);
```

```
                        }
                }

        # Initialize SSL for this connection
        if ($use_ssl) {
                $ssl_con = Net::SSLeay::new($ssl_ctx);
                Net::SSLeay::set_fd($ssl_con, fileno(SOCK));
                Net::SSLeay::accept($ssl_con) || exit;
                }

        # Work out the hostname for this web server
        $host = &get_socket_name(SOCK);
        $host || exit;
        $port = $config{'port'};
        $acptaddr = getpeername(SOCK);
        $acptaddr || exit;

        while(&handle_request($acptaddr, getsockname(SOCK))) { }
        close(SOCK);
        exit;
        }

# Build list of sockets to listen on
if ($config{"bind"} && $config{"bind"} ne "*") {
        push(@sockets, [ inet_aton($config{'bind'}), $config{'port'} ]);
        }
else {
        push(@sockets, [ INADDR_ANY, $config{'port'} ]);
        }
foreach $s (split(/\s+/, $config{'sockets'})) {
        if ($s =~ /^(\d+)$/) {
                # Just listen on another port on the main IP
                push(@sockets, [ $sockets[0]->[0], $s ]);
                }
        elsif ($s =~ /^(\S+):(\d+)$/) {
                # Listen on a specific port and IP
                push(@sockets, [ $1 eq "*" ? INADDR_ANY : inet_aton($1), $2 ]);
                }
        elsif ($s =~ /^([0-9\.]+):\*$/ || $s =~ /^([0-9\.]+)$/) {
                # Listen on the main port on another IP
                push(@sockets, [ inet_aton($1), $sockets[0]->[1] ]);
                }
        }

# Open all the sockets
$proto = getprotobyname('tcp');
for($i=0; $i<@sockets; $i++) {
        $fh = "MAIN$i";
        socket($fh, PF_INET, SOCK_STREAM, $proto) ||
                die "Failed to open socket : $!";
        setsockopt($fh, SOL_SOCKET, SO_REUSEADDR, pack("l", 1));
        for($j=0; $j<5; $j++) {
                last if (bind($fh, pack_sockaddr_in($sockets[$i]->[1],
                                                $sockets[$i]->[0])));
                sleep(1);
                }
        die "Failed to bind to $sockets[$i]->[1] : $!" if ($j == 5);
        listen($fh, SOMAXCONN);
        push(@socketfhs, $fh);
        }

if ($config{'listen'}) {
        # Open the socket that allows other webmin servers to find this one
        $proto = getprotobyname('udp');
        if (socket(LISTEN, PF_INET, SOCK_DGRAM, $proto)) {
                setsockopt(LISTEN, SOL_SOCKET, SO_REUSEADDR, pack("l", 1));
                bind(LISTEN, pack_sockaddr_in($config{'listen'}, INADDR_ANY));
                listen(LISTEN, SOMAXCONN);
                }
        else {
                $config{'listen'} = 0;
                }
        }

# Split from the controlling terminal
if (fork()) { exit; }
setsid();

# Close standard file handles
open(STDIN, "</dev/null");
open(STDOUT, ">/dev/null");
&log_error("miniserv.pl started");
&log_error($pam_msg) if ($pam_msg);

# write out the PID file
open(PIDFILE, "> $config{'pidfile'}");
printf PIDFILE "%d\n", getpid();
close(PIDFILE);

# Start the log-clearing process, if needed. This checks every minute
# to see if the log has passed its reset time, and if so clears it
if ($config{'logclear'}) {
        if (!($logclearer = fork())) {
                &close_all_sockets();
                close(LISTEN);
                while(1) {
                        local $write_logtime = 0;
                        local @st = stat("$config{'logfile'}.time");
                        if (@st) {
                                if ($st[9]+$config{'logtime'}*60*60 < time()){
                                        # need to clear log
                                        $write_logtime = 1;
                                        unlink($config{'logfile'});
                                        }
                                }
                        else { $write_logtime = 1; }
```

```
				if ($write_logtime) {
					open(LOGTIME, ">$config{'logfile'}.time");
					print LOGTIME time(),"\n";
					close(LOGTIME);
					}
				sleep(5*60);
				}
			exit;
			}
		push(@childpids, $logclearer);
		}

# Setup the logout time dbm if needed
if ($config{'session'}) {
		eval "use SDBM_File";
		dbmopen(%sessiondb, $config{'sessiondb'}, 0700);
		eval "\$sessiondb{'1111111111'} = 'foo bar';";
		if ($@) {
			dbmclose(%sessiondb);
			eval "use NDBM_File";
			dbmopen(%sessiondb, $config{'sessiondb'}, 0700);
			}
		else {
			delete($sessiondb{'1111111111'});
			}
		}

# Run the main loop
$SIG{'HUP'} = 'miniserv::trigger_restart';
$SIG{'TERM'} = 'miniserv::term_handler';
$SIG{'PIPE'} = 'IGNORE';
while(1) {
		# wait for a new connection, or a message from a child process
		local ($i, $rmask);
		if (@childpids <= $config{'maxconns'}) {
			# Only accept new main socket connects when ready
			local $s;
			foreach $s (@socketfhs) {
				vec($rmask, fileno($s), 1) = 1;
				}
			}
		else {
			printf STDERR "too many children (%d > %d)\n",
				scalar(@childpids), $config{'maxconns'};
			}
		if ($config{'passdelay'} || $config{'session'}) {
			for($i=0; $i<@passin; $i++) {
				vec($rmask, fileno($passin[$i]), 1) = 1;
				}
			}
		vec($rmask, fileno(LISTEN), 1) = 1 if ($config{'listen'});

		local $sel = select($rmask, undef, undef, 10);
```

```
if ($need_restart) { &restart_miniserv(); }
local $time_now = time();

# Clean up finished processes
local $pid;
do {	$pid = waitpid(-1, WNOHANG);
	@childpids = grep { $_ != $pid } @childpids;
	} while($pid > 0);

# run the unblocking procedure to check if enough time has passed to
# unblock hosts that heve been blocked because of password failures
if ($config{'blockhost_failures'}) {
	$i = 0;
	while ($i <= $#deny) {
		if ($blockhosttime{$deny[$i]} && $config{'blockhost_time'} != 0 &&
		   ($time_now - $blockhosttime{$deny[$i]}) >= $config{'blockhost_time'}) {
			# the host can be unblocked now
			$hostfail{$deny[$i]} = 0;
			splice(@deny, $i, 1);
			}
		$i++;
		}
	}

if ($config{'session'} && (++$remove_session_count%50) == 0) {
	# Remove sessions with more than 7 days of inactivity,
	local $s;
	foreach $s (keys %sessiondb) {
		local ($user, $ltime) = split(/\s+/, $sessiondb{$s});
		if ($time_now - $ltime > 7*24*60*60) {
			local @sdb = split(/\s+/, $sessiondb{$s});
			&run_logout_script($s, $sdb[0]);
			delete($sessiondb{$s});
			if ($use_syslog) {
				syslog("info", "Timeout of $sdb[0]");
				}
			}
		}
	}
next if ($sel <= 0);

# Check if any of the main sockets have received a new connection
local $sn = 0;
foreach $s (@socketfhs) {
	if (vec($rmask, fileno($s), 1)) {
		# got new connection
		$acptaddr = accept(SOCK, $s);
		if (!$acptaddr) { next; }
		binmode(SOCK);# turn off any Perl IO stuff

		# create pipes
		local ($PASSINr, $PASSINw, $PASSOUTr, $PASSOUTw);
		if ($config{'passdelay'} || $config{'session'}) {
```

```
		local $p;
		local %taken = map { $_, 1 } @passin;
		for($p=0; $taken{"PASSINr$p"}; $p++) { }
		$PASSINr = "PASSINr$p";
		$PASSINw = "PASSINw$p";
		$PASSOUTr = "PASSOUTr$p";
		$PASSOUTw = "PASSOUTw$p";
		pipe($PASSINr, $PASSINw);
		pipe($PASSOUTr, $PASSOUTw);
		select($PASSINw); $| = 1;
		select($PASSINr); $| = 1;
		select($PASSOUTw); $| = 1;
		select($PASSOUTw); $| = 1;
		}
	select(STDOUT);

	# Check username of connecting user
	local ($peerp, $peera) = unpack_sockaddr_in($acptaddr);
	$localauth_user = undef;
	if ($config{'localauth'} && inet_ntoa($peera) eq "127.0.0.1") {
		if (open(TCP, "/proc/net/tcp")) {
			# Get the info direct from the kernel
			while(<TCP>) {
				s/^\s+//;
				local @t = split(/[\s:]+/, $_);
				if ($t[1] eq '0100007F' &&
				   $t[2] eq sprintf("%4.4X", $peerp)) {
					$localauth_user = getpwuid($t[11]);
					last;
					}
				}
			close(TCP);
			}
		else {
			# Call lsof for the info
			local $lsofpid = open(LSOF,
				"$config{'localauth'} -i TCP\@127.0.0.1:$peerp |");
			while(<LSOF>) {
				if (/^(\S+)\s+(\d+)\s+(\S+)/ &&
				   $2 != $$ && $2 != $lsofpid) {
					$localauth_user = $3;
					}
				}
			close(LSOF);
			}
		}

	# Work out the hostname for this web server
	$host = &get_socket_name(SOCK);
	if (!$host) {
		print STDERR "Failed to get local socket name : $!\n";
		close(SOCK);
		next;
```

```
				}
			$port = $sockets[$sn]->[1];

			# fork the subprocess
			local $handpid;
			if (!($handpid = fork())) {
				# setup signal handlers
				$SIG{'TERM'} = 'DEFAULT';
				$SIG{'PIPE'} = 'DEFAULT';
				#$SIG{'CHLD'} = 'IGNORE';
				$SIG{'HUP'} = 'IGNORE';

				# Initialize SSL for this connection
				if ($use_ssl) {
					$ssl_con = Net::SSLeay::new($ssl_ctx);
					Net::SSLeay::set_fd($ssl_con, fileno(SOCK));
					Net::SSLeay::accept($ssl_con) || exit;
					}

				# close useless pipes
				if ($config{'passdelay'} || $config{'session'}) {
					local $p;
					foreach $p (@passin) { close($p); }
					foreach $p (@passout) { close($p); }
					close($PASSINr); close($PASSOUTw);
					}
				&close_all_sockets();
				close(LISTEN);

				while(&handle_request($acptaddr, getsockname(SOCK))) { }
				shutdown(SOCK, 1);
				close(SOCK);
				close($PASSINw); close($PASSOUTw);
				exit;
				}
			push(@childpids, $handpid);
			if ($config{'passdelay'} || $config{'session'}) {
				close($PASSINw); close($PASSOUTr);
				push(@passin, $PASSINr); push(@passout, $PASSOUTw);
				}
			close(SOCK);
			}
		$sn++;
		}

	if ($config{'listen'} && vec($rmask, fileno(LISTEN), 1)) {
		# Got UDP packet from another webmin server
		local $rcvbuf;
		local $from = recv(LISTEN, $rcvbuf, 1024, 0);
		next if (!$from);
		local $fromip = inet_ntoa((unpack_sockaddr_in($from))[1]);
		local $toip = inet_ntoa((unpack_sockaddr_in(
					getsockname(LISTEN)))[1]);
		if ((!@deny || !&ip_match($fromip, $toip, @deny)) &&
		  (!@allow || &ip_match($fromip, $toip, @allow))) {
			local $listenhost = &get_socket_name(LISTEN);
			send(LISTEN, "$listenhost:$config{'port'}:".
				     ($use_ssl || $config{'inetd_ssl'} ? 1 : 0),
				     0, $from)
				if ($listenhost);
			}
		}

	# check for password-timeout messages from subprocesses
	for($i=0; $i<@passin; $i++) {
		if (vec($rmask, fileno($passin[$i]), 1)) {
			# this sub-process is asking about a password
			local $infd = $passin[$i];
			local $outfd = $passout[$i];
			local $inline = <$infd>;
			if ($inline =~ /^delay\s+(\S+)\s+(\S+)\s+(\d+)/) {
				# Got a delay request from a subprocess.. for
				# valid logins, there is no delay (to prevent
				# denial of service attacks), but for invalid
				# logins the delay increases with each failed
				# attempt.
				if ($3) {
					# login OK.. no delay
					print $outfd "0 0\n";
					$hostfail{$2} = 0;
					}
				else {
					# login failed..
					$hostfail{$2}++;
					# add the host to the block list if necessary
					if ($config{'blockhost_failures'} &&
					   $hostfail{$2} >= $config{'blockhost_failures'}) {
						push(@deny, $2);
						$blockhosttime{$2} = $time_now;
						$blocked = 1;
						if ($use_syslog) {
							local $logtext = "Security alert: Host $2 ".
							  "blocked after
$config{'blockhost_failures'} ".
							  "failed logins for user $1";
							syslog("crit", $logtext);
							}
						}
					else {
						$blocked = 0;
						}
					$dl = $userdlay{$1} -
					    int(($time_now - $userlast{$1})/50);
					$dl = $dl < 0 ? 0 : $dl+1;
					print $outfd "$dl $blocked\n";
					$userdlay{$1} = $dl;
```

```
                                                }
                                        $userlast{$1} = $time_now;
                                        }
                                elsif ($inline =~ /^verify\s+(\S+)/) {
                                        # Verifying a session ID
                                        local $session_id = $1;
                                        if (!defined($sessiondb{$session_id})) {
                                                # Session doesn't exist
                                                print $outfd "0 0\n";
                                                }
                                        else {
                                                local ($user, $ltime) = split(/\s+/, $sessiondb{$session_id});
                                                if ($config{'logouttime'} &&
                                                   $time_now - $ltime > $config{'logouttime'}*60) {
                                                        # Session has timed out
                                                        print $outfd "1 ",$time_now - $ltime,"\n";
                                                        #delete($sessiondb{$session_id});
                                                        }
                                                else {
                                                        # Session is OK
                                                        print $outfd "2 $user\n";
                                                        if ($config{'logouttime'} &&
                                                           $time_now - $ltime >
($config{'logouttime'}*60)/2) {
                                                                $sessiondb{$session_id} = "$user
$time_now";
                                                                }
                                                        }
                                                }
                                        }
                                elsif ($inline =~ /^new\s+(\S+)\s+(\S+)/) {
                                        # Creating a new session
                                        $sessiondb{$1} = "$2 $time_now";
                                        }
                                elsif ($inline =~ /^delete\s+(\S+)/) {
                                        # Logging out a session
                                        local $sid = $1;
                                        local @sdb = split(/\s+/, $sessiondb{$sid});
                                        print $outfd $sdb[0],"\n";
                                        delete($sessiondb{$sid});
                                        }
                                else {
                                        # close pipe
                                        close($infd); close($outfd);
                                        $passin[$i] = $passout[$i] = undef;
                                        }
                                }
                        }
        @passin = grep { defined($_) } @passin;
        @passout = grep { defined($_) } @passout;
        }

# handle_request(remoteaddress, localaddress)
# Where the real work is done
sub handle_request
{
$acptip = inet_ntoa((unpack_sockaddr_in($_[0]))[1]);
$localip = $_[1] ? inet_ntoa((unpack_sockaddr_in($_[1]))[1]) : undef;
if ($config{'loghost'}) {
	$acpthost = gethostbyaddr(inet_aton($acptip), AF_INET);
	$acpthost = $acptip if (!$acpthost);
	}
else {
	$acpthost = $acptip;
	}
$datestr = &http_date(time());
$ok_code = 200;
$ok_message = "Document follows";
$logged_code = undef;
$reqline = $request_uri = $page = undef;

# Wait at most 60 secs for start of headers for initial requests, or
# 10 minutes for kept-alive connections
local $rmask;
vec($rmask, fileno(SOCK), 1) = 1;
local $sel = select($rmask, undef, undef, $checked_timeout ? 10*60 : 60);
if (!$sel) {
	if ($checked_timeout) { exit; }
	else { &http_error(400, "Timeout"); }
	}
$checked_timeout++;

# Read the HTTP request and headers
local $origreqline = &read_line();
($reqline = $origreqline) =~ s/\r|\n//g;
$method = $page = $request_uri = undef;
if (!$reqline && (!$use_ssl || $checked_timeout > 1)) {
	# An empty request .. just close the connection
	return 0;
	}
elsif ($reqline !~ /^(GET|POST|HEAD)\s+(.*)\s+HTTP\/1\..$/) {
	if ($use_ssl) {
		# This could be an http request when it should be https
		$use_ssl = 0;
		local $url = "https://$host:$port/";
		if ($config{'ssl_redirect'}) {
			# Just re-direct to the correct URL
			&write_data("HTTP/1.0 302 Moved Temporarily\r\n");
			&write_data("Date: $datestr\r\n");
			&write_data("Server: $config{'server'}\r\n");
			&write_data("Location: $url\r\n");
			&write_keep_alive(0);
			&write_data("\r\n");
			return 0;
			}
		else {
```

```
                        # Tell user the correct URL
                        &http_error(200, "Bad Request", "This web server is running in SSL mode. Try
the URL <a href='$url'>$url</a> instead.<br>");
                        }
                }
        elsif (ord(substr($reqline, 0, 1)) == 128 && !$use_ssl) {
                # This could be an https request when it should be http ..
                # need to fake a HTTP response
                eval <<'EOF';
                        use Net::SSLeay;
                        eval "Net::SSLeay::SSLeay_add_ssl_algorithms()";
                        eval "Net::SSLeay::load_error_strings()";
                        $ssl_ctx = Net::SSLeay::CTX_new();
                        Net::SSLeay::CTX_use_RSAPrivateKey_file(
                                $ssl_ctx, $config{'keyfile'},
                                &Net::SSLeay::FILETYPE_PEM);
                        Net::SSLeay::CTX_use_certificate_file(
                                $ssl_ctx,
                                $config{'certfile'} || $config{'keyfile'},
                                &Net::SSLeay::FILETYPE_PEM);
                        $ssl_con = Net::SSLeay::new($ssl_ctx);
                        pipe(SSLr, SSLw);
                        if (!fork()) {
                                close(SSLr);
                                select(SSLw); $| = 1; select(STDOUT);
                                print SSLw $origreqline;
                                local $buf;
                                while(sysread(SOCK, $buf, 1) > 0) {
                                        print SSLw $buf;
                                        }
                                close(SOCK);
                                exit;
                                }
                        close(SSLw);
                        Net::SSLeay::set_wfd($ssl_con, fileno(SOCK));
                        Net::SSLeay::set_rfd($ssl_con, fileno(SSLr));
                        Net::SSLeay::accept($ssl_con) || die "accept() failed";
                        $use_ssl = 1;
                        local $url = "http://$host:$port/";
                        if ($config{'ssl_redirect'}) {
                                # Just re-direct to the correct URL
                                &write_data("HTTP/1.0 302 Moved Temporarily\r\n");
                                &write_data("Date: $datestr\r\n");
                                &write_data("Server: $config{'server'}\r\n");
                                &write_data("Location: $url\r\n");
                                &write_keep_alive(0);
                                &write_data("\r\n");
                                return 0;
                                }
                        else {
                                # Tell user the correct URL
                                &http_error(200, "Bad Request", "This web server is not running in
SSL mode. Try the URL <a href='$url'>$url</a> instead.<br>");
```

```
                                        }
EOF
                        if ($@) {
                                &http_error(400, "Bad Request");
                                }
                        }
                else {
                        &http_error(400, "Bad Request");
                        }
                }
$method = $1;
$request_uri = $page = $2;
%header = ();
local $lastheader;
while(1) {
                ($headline = &read_line()) =~ s/\r|\n//g;
                last if ($headline eq "");
                if ($headline =~ /^(\S+):\s*(.*)$/) {
                        $header{$lastheader = lc($1)} = $2;
                        }
                elsif ($headline =~ /^\s+(.*)$/) {
                        $header{$lastheader} .= $headline;
                        }
                else {
                        &http_error(400, "Bad Header $headline");
                        }
                }
if (defined($header{'host'})) {
                if ($header{'host'} =~ /^([^:]+):([0-9]+)$/) { $host = $1; $port = $2; }
                else { $host = $header{'host'}; }
                if ($config{'musthost'} && $host ne $config{'musthost'}) {
                        # Disallowed hostname used
                        &http_error(400, "Invalid HTTP hostname");
                        }
                }
undef(%in);
if ($page =~ /^([^\?]+)\?(.*)$/) {
                # There is some query string information
                $page = $1;
                $querystring = $2;
                if ($querystring !~ /=/) {
                        $queryargs = $querystring;
                        $queryargs =~ s/\+/ /g;
                        $queryargs =~ s/%(..)/pack("c",hex($1))/ge;
                        $querystring = "";
                        }
                else {
                        # Parse query-string parameters
                        local @in = split(/\&/, $querystring);
                        foreach $i (@in) {
                                local ($k, $v) = split(/=/, $i, 2);
                                $k =~ s/\+/ /g; $k =~ s/%(..)/pack("c",hex($1))/ge;
                                $v =~ s/\+/ /g; $v =~ s/%(..)/pack("c",hex($1))/ge;
                        $in{$k} = $v;
                        }
                }
        }
$posted_data = undef;
if ($method eq 'POST' &&
   $header{'content-type'} eq 'application/x-www-form-urlencoded') {
        # Read in posted query string information, up the configured maximum
        # post request length
        $clen = $header{"content-length"};
        $clen_read = $clen > $config{'max_post'} ? $config{'max_post'} : $clen;
        while(length($posted_data) < $clen_read) {
                $buf = &read_data($clen_read - length($posted_data));
                if (!length($buf)) {
                        &http_error(500, "Failed to read POST request");
                        }
                chomp($posted_data);
                $posted_data =~ s/\015$//mg;
                $posted_data .= $buf;
                }
        if ($clen_read != $clen) {
                # If the client sent more data than we asked for, chop the
                # rest off
                $posted_data = substr($posted_data, 0, $clen)
                        if (length($posted_data) > $clen);
                }
        #$posted_data =~ s/\r|\n//g;        # some browsers include an extra newline
        #                                   # in the data!
        local @in = split(/\&/, $posted_data);
        foreach $i (@in) {
                local ($k, $v) = split(/=/, $i, 2);
                $k =~ s/\+/ /g; $k =~ s/%(..)/pack("c",hex($1))/ge;
                $v =~ s/\+/ /g; $v =~ s/%(..)/pack("c",hex($1))/ge;
                $in{$k} = $v;
                }
        }

# replace %XX sequences in page
$page =~ s/%(..)/pack("c",hex($1))/ge;

# check address against access list
if (@deny && &ip_match($acptip, $localip, @deny) ||
   @allow && !&ip_match($acptip, $localip, @allow)) {
        &http_error(403, "Access denied for $acptip");
        return 0;
        }

if ($use_libwrap) {
        # Check address with TCP-wrappers
        if (!hosts_ctl($config{'pam'}, STRING_UNKNOWN, $acptip, STRING_UNKNOWN)) {
                &http_error(403, "Access denied for $acptip");
                return 0;
                }
```

```
	}

# check for the logout flag file, and if existant deny authentication
if ($config{'logout'} && -r $config{'logout'}.$in{'miniserv_logout_id'}) {
	$deny_authentication++;
	open(LOGOUT, $config{'logout'}.$in{'miniserv_logout_id'});
	chop($count = <LOGOUT>);
	close(LOGOUT);
	$count--;
	if ($count > 0) {
		open(LOGOUT, ">$config{'logout'}$in{'miniserv_logout_id'}");
		print LOGOUT "$count\n";
		close(LOGOUT);
		}
	else {
		unlink($config{'logout'}.$in{'miniserv_logout_id'});
		}
	}

# check for any redirect for the requested URL
$simple = &simplify_path($page, $bogus);
$rpath = $simple;
$rpath .= "&".$querystring if (defined($querystring));
$redir = $redirect{$rpath};
if (defined($redir)) {
	&write_data("HTTP/1.0 302 Moved Temporarily\r\n");
	&write_data("Date: $datestr\r\n");
	&write_data("Server: $config{'server'}\r\n");
	local $ssl = $use_ssl || $config{'inetd_ssl'};
	$portstr = $port == 80 && !$ssl ? "" :
		     $port == 443 && $ssl ? "" : ":$port";
	$prot = $ssl ? "https" : "http";
	&write_data("Location: $prot://$host$portstr$redir\r\n");
	&write_keep_alive(0);
	&write_data("\r\n");
	return 0;
	}

# Check for password if needed
if (%users) {
	$validated = 0;
	$blocked = 0;

	# Session authentication is never used for connections by
	# another webmin server
	if ($header{'user-agent'} =~ /webmin/i) {
		$config{'session'} = 0;
		}

	# check for SSL authentication
	if ($use_ssl && $verified_client) {
		$peername = Net::SSLeay::X509_NAME_oneline(
					Net::SSLeay::X509_get_subject_name(
                                        Net::SSLeay::get_peer_certificate(
                                                $ssl_con)));
                foreach $u (keys %certs) {
                        if ($certs{$u} eq $peername) {
                                $authuser = $u;
                                $validated = 2;
                                #syslog("info", "SSL login as $authuser from $acpthost") if
($use_syslog);
                                last;
                                }
                        }
                if ($use_syslog && !$validated) {
                        syslog("crit",
                            "Unknown SSL certificate $peername");
                        }
                }

        # Check for normal HTTP authentication
        if (!$validated && !$deny_authentication && !$config{'session'} &&
          $header{authorization} =~ /^basic\s+(\S+)$/i) {
                # authorization given..
                ($authuser, $authpass) = split(/:/, &b64decode($1), 2);
                local ($vu, $expired, $nonexist) =
                        &validate_user($authuser, $authpass);
                if ($vu && (!$expired || $config{'passwd_mode'} == 1)) {
                        $authuser = $vu;
                        $validated = 1;
                        }
                else {
                        $validated = 0;
                        }
                if ($use_syslog && !$validated) {
                        syslog("crit",
                            ($nonexist ? "Non-existent" :
                                $expired ? "Expired" : "Invalid").
                            " login as $authuser from $acpthost");
                        }
                if ($authuser =~ /\r|\n|\s/) {
                        &http_error(500, "Invalid username",
                                  "Username contains invalid characters");
                        }
                if ($authpass =~ /\r|\n/) {
                        &http_error(500, "Invalid password",
                                  "Password contains invalid characters");
                        }

                if ($config{'passdelay'} && !$config{'inetd'}) {
                        # check with main process for delay
                        print $PASSINw "delay $authuser $acptip $validated\n";
                        <$PASSOUTr> =~ /(\d+) (\d+)/;
                        $blocked = $2;
                        sleep($1);
                        }
        }

    # Check for a visit to the special session login page
    if ($config{'session'} && !$deny_authentication &&
       $page eq $config{'session_login'}) {
            if ($in{'logout'} && $header{'cookie'} =~ /(^|\s)$sidname=([a-f0-9]+)/) {
                    # Logout clicked .. remove the session
                    local $sid = $2;
                    print $PASSINw "delete $sid\n";
                    local $louser = <$PASSOUTr>;
                    chop($louser);
                    $logout = 1;
                    $already_session_id = undef;
                    $authuser = $baseauthuser = undef;
                    if ($louser) {
                            if ($use_syslog) {
                                    syslog("info", "Logout by $louser from $acpthost");
                                    }
                            &run_logout_script($louser, $sid,
                                                    $acptip, $localip);
                            }
                    }
            else {
                    # Validate the user
                    if ($in{'user'} =~ /\r|\n|\s/) {
                            &http_error(500, "Invalid username",
                               "Username contains invalid characters");
                            }
                    if ($in{'pass'} =~ /\r|\n/) {
                            &http_error(500, "Invalid password",
                               "Password contains invalid characters");
                            }

                    local ($vu, $expired, $nonexist) =
                            &validate_user($in{'user'}, $in{'pass'});
                    local $ok = $vu ? 1 : 0;
                    $authuser = $vu if ($vu);
                    local $loginuser = $authuser || $in{'user'};

                    # check if the test cookie is set
                    if ($header{'cookie'} !~ /testing=1/ && $loginuser &&
                       !$config{'no_testing_cookie'}) {
                            &http_error(500, "No cookies",
                              "Your browser does not support cookies, ".
                              "which are required for this web server to ".
                              "work in session authentication mode");
                            }

                    # check with main process for delay
                    if ($config{'passdelay'} && $loginuser) {
                            print $PASSINw "delay $loginuser $acptip $ok\n";
                            <$PASSOUTr> =~ /(\d+) (\d+)/;
                            $blocked = $2;
```

```
                                        sleep($1);
                                        }

                        if ($ok && (!$expired ||
                                $config{'passwd_mode'} == 1)) {
                                        # Logged in OK! Tell the main process about
                                        # the new SID
                                        local $sid;
                                        $SIG{ALRM} = "miniserv::urandom_timeout";
                                        alarm(5);
                                        if (open(RANDOM, "/dev/urandom")) {
                                                my $tmpsid;
                                                if (read(RANDOM, $tmpsid, 16) == 16) {
                                                        $sid = lc(unpack('h*',$tmpsid));
                                                        }
                                                close(RANDOM);
                                                }
                                        alarm(0);
                                        if (!$sid) {
                                                $sid = time();
                                                local $mul = 1;
                                                foreach $c (split(//, crypt($in{'pass'}, substr($$, -2)))) {
                                                        $sid += ord($c) * $mul;
                                                        $mul *= 3;
                                                        }
                                                }

                                        print $PASSINw "new $sid $authuser\n";

                                        # Run the post-login script, if any
                                        &run_login_script($authuser, $sid,
                                                          $acptip, $localip);

                                        # Set cookie and redirect
                                        &write_data("HTTP/1.0 302 Moved Temporarily\r\n");
                                        &write_data("Date: $datestr\r\n");
                                        &write_data("Server: $config{'server'}\r\n");
                                        local $ssl = $use_ssl || $config{'inetd_ssl'};
                                        $portstr = $port == 80 && !$ssl ? "" :
                                                   $port == 443 && $ssl ? "" : ":$port";
                                        $prot = $ssl ? "https" : "http";
                                        local $sec = $ssl ? "; secure" : "";
                                        #$sec .= "; httpOnly";
                                        if ($in{'save'}) {
                                                &write_data("Set-Cookie: $sidname=$sid; path=/;
expires=\"Fri, 1-Jan-2038 00:00:01\"$sec\r\n");
                                                }
                                        else {
                                                &write_data("Set-Cookie: $sidname=$sid; path=/$sec\r\n");
                                                }
                                        &write_data("Location: $prot://$host$portstr$in{'page'}\r\n");
                                        &write_keep_alive(0);
                                        &write_data("\r\n");
                                &log_request($acpthost, $authuser, $reqline, 302, 0);
                                syslog("info", "Successful login as $authuser from $acpthost") if
($use_syslog);
                                return 0;
                                }
                        elsif ($ok && $expired &&
                            $config{'passwd_mode'} == 2) {
                                # Login was ok, but password has expired. Need
                                # to force display, of password change form.
                                $validated = 1;
                                $authuser = undef;
                                $querystring = "&user=".&urlize($vu);
                                $method = "GET";
                                $queryargs = "";
                                $page = $config{'password_form'};
                                $logged_code = 401;
                                $miniserv_internal = 2;
                                }
                        else {
                                # Login failed, or password has expired. Display
                                # the form again.
                                $failed_user = $in{'user'};
                                $request_uri = $in{'page'};
                                $already_session_id = undef;
                                $method = "GET";
                                $authuser = $baseauthuser = undef;
                                syslog("crit",
                                        ($nonexist ? "Non-existent" :
                                         $expired ? "Expired" : "Invalid").
                                        " login as $in{'user'} from $acpthost")
                                        if ($use_syslog);
                                }
                        }
                }

        # Check for a visit to the special password change page
        if ($config{'session'} && !$deny_authentication &&
           $page eq $config{'password_change'} && !$validated &&
           $config{'passwd_mode'} == 2) {
                # Just let this slide ..
                $validated = 1;
                $miniserv_internal = 3;
                }

        # Check for an existing session
        if ($config{'session'} && !$validated) {
                if ($already_session_id) {
                        $session_id = $already_session_id;
                        $authuser = $already_authuser;
                        $validated = 1;
                        }
                elsif (!$deny_authentication &&
                    $header{'cookie'} =~ /(^|\s)$sidname=([a-f0-9]+)/) {
```

```
				$session_id = $2;
				print $PASSINw "verify $session_id\n";
				<$PASSOUTr> =~ /(\d+)\s+(\S+)/;
				if ($1 == 2) {
					# Valid session continuation
					$validated = 1;
					$authuser = $2;
					#$already_session_id = $session_id;
					$already_authuser = $authuser;
					}
				elsif ($1 == 1) {
					# Session timed out
					$timed_out = $2;
					}
				else {
					# Invalid session ID .. don't set verified
					}
				}
			}

	# Check for local authentication
	if ($localauth_user && !$header{'x-forwarded-for'} && !$header{'via'}) {
		if (defined($users{$localauth_user})) {
			# Local user exists in webmin users file
			$validated = 1;
			$authuser = $localauth_user;
			# syslog("info", "Local login as $authuser from $acpthost") if ($use_syslog);
			}
		elsif ($config{'unixauth'}) {
			# Local user must exist
			$validated = 2;
			$authuser = $localauth_user;
			# syslog("info", "Local login as $authuser from $acpthost") if ($use_syslog);
			}
		else {
			$localauth_user = undef;
			}
		}

	if (!$validated) {
		# Check if this path allows anonymous access
		local $a;
		foreach $a (keys %anonymous) {
			if (substr($simple, 0, length($a)) eq $a) {
				# It does! Auth as the user ..
				$validated = 3;
				$baseauthuser = $authuser = $anonymous{$a};
				}
			}
		}

	if (!$validated) {
		# Check if this path allows unauthenticated access
		local ($u, $unauth);
		foreach $u (@unauth) {
			$unauth++ if ($simple =~ /$u/);
			}
		if (!$bogus && $unauth) {
			# Unauthenticated directory or file request - approve it
			$validated = 3;
			$baseauthuser = $authuser = undef;
			}
		}

	if (!$validated) {
		if ($blocked == 0) {
			# No password given.. ask
			if ($config{'session'}) {
				# Force CGI for session login
				$validated = 1;
				if ($logout) {
					$querystring .= "&logout=1&page=/";
					}
				else {
					# Re-direct to current module only
					local $rpage = $request_uri;
					$rpage =~ s/[^\/]+$//
						if (!$config{'loginkeeppage'});
					$querystring = "page=".&urlize($rpage);
					}
				$method = "GET";
				$querystring .= "&failed=$failed_user" if ($failed_user);
				$querystring .= "&timed_out=$timed_out" if ($timed_out);
				$queryargs = "";
				$page = $config{'session_login'};
				$miniserv_internal = 1;
				$logged_code = 401;
				}
			else {
				# Ask for login with HTTP authentication
				&write_data("HTTP/1.0 401 Unauthorized\r\n");
				&write_data("Date: $datestr\r\n");
				&write_data("Server: $config{'server'}\r\n");
				&write_data("WWW-authenticate: Basic ".
					"realm=\"$config{'realm'}\"\r\n");
				&write_keep_alive(0);
				&write_data("Content-type: text/html\r\n");
				&write_data("\r\n");
				&reset_byte_count();
				&write_data("<html>\n");
				&write_data("<head><title>Unauthorized</title></head>\n");
				&write_data("<body><h1>Unauthorized</h1>\n");
				&write_data("A password is required to access this\n");
				&write_data("web server. Please try again. <p>\n");
				&write_data("</body></html>\n");
				&log_request($acpthost, undef, $reqline, 401, &byte_count());
```

```
                                        return 0;
                                        }
                                }
                        else {
                                # when the host has been blocked, give it an error message
                                &http_error(403, "Access denied for $acptip. The host has been blocked "
                                        ."because of too many authentication failures.");
                                }
                        }
                else {
                        # If we are using unixauth, keep the 'real' username
                        if ($config{'unixauth'} && !$users{$authuser}) {
                                $baseauthuser = $config{'unixauth'};
                                }
                        else {
                                $baseauthuser = $authuser;
                                }

                        if ($config{'remoteuser'} && !$< && $validated) {
                                # Switch to the UID of the remote user (if he exists)
                                local @u = getpwnam($authuser);
                                if (@u && $< != $u[2]) {
                                        $( = $u[3]; $) = "$u[3] $u[3]";
                                        ($>, $<) = ($u[2], $u[2]);
                                        }
                                else {
                                        &http_error(500, "Unix user $authuser does not exist");
                                        return 0;
                                        }
                                }
                        }

                # Check per-user IP access control
                if ($deny{$baseauthuser} && &ip_match($acptip, $localip, @{$deny{$baseauthuser}}) ||
                   $allow{$baseauthuser} && !&ip_match($acptip, $localip, @{$allow{$baseauthuser}})) {
                        &http_error(403, "Access denied for $acptip");
                        return 0;
                        }
                }

# Figure out what kind of page was requested
rerun:
$simple = &simplify_path($page, $bogus);
$simple =~ s/[\000-\037]//g;
if ($bogus) {
        &http_error(400, "Invalid path");
        }
local ($full, @stfull);
local $preroot = $authuser && defined($config{'preroot_'.$authuser}) ?
                        $config{'preroot_'.$authuser} :
                $authuser && $baseauthuser && defined($config{'preroot_'.$baseauthuser}) ?
                        $config{'preroot_'.$baseauthuser} :
                        $config{'preroot'};
```

```
if ($preroot) {
        # Always under the current webmin root
        $preroot =~ s/^.*\/\//g;
        $preroot = $config{'root'}.'/'.$preroot;
        }
if ($preroot) {
        # Look in the template root directory first
        $is_directory = 1;
        $sofar = "";
        $full = $preroot.$sofar;
        $scriptname = $simple;
        foreach $b (split(/\//, $simple)) {
                if ($b ne "") { $sofar .= "/$b"; }
                $full = $preroot.$sofar;
                @stfull = stat($full);
                if (!@stfull) { undef($full); last; }

                # Check if this is a directory
                if (-d _) {
                        # It is.. go on parsing
                        $is_directory = 1;
                        next;
                        }
                else { $is_directory = 0; }

                # Check if this is a CGI program
                if (&get_type($full) eq "internal/cgi") {
                        $pathinfo = substr($simple, length($sofar));
                        $pathinfo .= "/" if ($page =~ /\/$/);
                        $scriptname = $sofar;
                        last;
                        }
                }
        if ($full) {
                if ($sofar eq '') {
                        $cgi_pwd = $config{'root'};
                        }
                elsif ($is_directory) {
                        $cgi_pwd = "$config{'root'}$sofar";
                        }
                else {
                        "$config{'root'}$sofar" =~ /^(.*\/)[^\/]+$/;
                        $cgi_pwd = $1;
                        }
                if ($is_directory) {
                        # Check for index files in the directory
                        local $foundidx;
                        foreach $idx (split(/\s+/, $config{"index_docs"})) {
                                $idxfull = "$full/$idx";
                                local @stidxfull = stat($idxfull);
                                if (-r _ && !-d _) {
                                        $full = $idxfull;
                                        @stfull = @stidxfull;
```

```
                    $is_directory = 0;
                    $scriptname .= "/"
                        if ($scriptname ne "/");
                    $foundidx++;
                    last;
                    }
                }
            @stfull = stat($full) if (!$foundidx);
            }
        }
    }
if (!$full || $is_directory) {
    $sofar = "";
    $full = $config{"root"} . $sofar;
    $scriptname = $simple;
    foreach $b (split(/\//, $simple)) {
        if ($b ne "") { $sofar .= "/$b"; }
        $full = $config{"root"} . $sofar;
        @stfull = stat($full);
        if (!@stfull) { &http_error(404, "File not found"); }

        # Check if this is a directory
        if (-d _) {
            # It is.. go on parsing
            next;
            }

        # Check if this is a CGI program
        if (&get_type($full) eq "internal/cgi") {
            $pathinfo = substr($simple, length($sofar));
            $pathinfo .= "/" if ($page =~ /\/$/);
            $scriptname = $sofar;
            last;
            }
        }
    $full =~ /^(.*\/)[^\/]+$/; $cgi_pwd = $1;
    }
@stfull = stat($full) if (!@stfull);

# check filename against denyfile regexp
local $denyfile = $config{'denyfile'};
if ($denyfile && $full =~ /$denyfile/) {
    &http_error(403, "Access denied to $page");
    return 0;
    }

# Reached the end of the path OK.. see what we've got
if (-d _) {
    # See if the URL ends with a / as it should
    if ($page !~ /\/$/) {
        # It doesn't.. redirect
        &write_data("HTTP/1.0 302 Moved Temporarily\r\n");
        $ssl = $use_ssl || $config{'inetd_ssl'};
```

```
                $portstr = $port == 80 && !$ssl ? "" :
                        $port == 443 && $ssl ? "" : ":$port";
                &write_data("Date: $datestr\r\n");
                &write_data("Server: $config{server}\r\n");
                $prot = $ssl ? "https" : "http";
                &write_data("Location: $prot://$host$portstr$page/\r\n");
                &write_keep_alive(0);
                &write_data("\r\n");
                &log_request($acpthost, $authuser, $reqline, 302, 0);
                return 0;
                }
        # A directory.. check for index files
        local $foundidx;
        foreach $idx (split(/\s+/, $config{"index_docs"})) {
                $idxfull = "$full/$idx";
                @stidxfull = stat($idxfull);
                if (-r _ && !-d _) {
                        $cgi_pwd = $full;
                        $full = $idxfull;
                        @stfull = @stidxfull;
                        $scriptname .= "/" if ($scriptname ne "/");
                        $foundidx++;
                        last;
                        }
                }
        @stfull = stat($full) if (!$foundidx);
        }
if (-d _) {
        # This is definately a directory.. list it
        &write_data("HTTP/1.0 $ok_code $ok_message\r\n");
        &write_data("Date: $datestr\r\n");
        &write_data("Server: $config{server}\r\n");
        &write_data("Content-type: text/html\r\n");
        &write_keep_alive(0);
        &write_data("\r\n");
        &reset_byte_count();
        &write_data("<h1>Index of $simple</h1>\n");
        &write_data("<pre>\n");
        &write_data(sprintf "%-35.35s %-20.20s %-10.10s\n",
                        "Name", "Last Modified", "Size");
        &write_data("<hr>\n");
        opendir(DIR, $full);
        while($df = readdir(DIR)) {
                if ($df =~ /^\./) { next; }
                (@stbuf = stat("$full/$df")) || next;
                if (-d _) { $df .= "/"; }
                @tm = localtime($stbuf[9]);
                $fdate = sprintf "%2.2d/%2.2d/%4.4d %2.2d:%2.2d:%2.2d",
                                $tm[3],$tm[4]+1,$tm[5]+1900,
                                $tm[0],$tm[1],$tm[2];
                $len = length($df); $rest = " "x(35-$len);
                &write_data(sprintf
                 "<a href=\"%s\">%-${len}.${len}s</a>$rest %-20.20s %-10.10s\n",
                $df, $df, $fdate, $stbuf[7]);
                }
        closedir(DIR);
        &log_request($acpthost, $authuser, $reqline, $ok_code, &byte_count());
        return 0;
        }

# CGI or normal file
local $rv;
if (&get_type($full) eq "internal/cgi") {
        # A CGI program to execute
        $envtz = $ENV{"TZ"};
        $envuser = $ENV{"USER"};
        $envpath = $ENV{"PATH"};
        $envlang = $ENV{"LANG"};
        foreach (keys %ENV) { delete($ENV{$_}); }
        $ENV{"PATH"} = $envpath if ($envpath);
        $ENV{"TZ"} = $envtz if ($envtz);
        $ENV{"USER"} = $envuser if ($envuser);
        $ENV{"OLD_LANG"} = $envlang if ($envlang);
        $ENV{"HOME"} = $user_homedir;
        $ENV{"SERVER_SOFTWARE"} = $config{"server"};
        $ENV{"SERVER_NAME"} = $host;
        $ENV{"SERVER_ADMIN"} = $config{"email"};
        $ENV{"SERVER_ROOT"} = $config{"root"};
        $ENV{"SERVER_PORT"} = $port;
        $ENV{"REMOTE_HOST"} = $acpthost;
        $ENV{"REMOTE_ADDR"} = $acptip;
        $ENV{"REMOTE_USER"} = $authuser if (defined($authuser));
        $ENV{"BASE_REMOTE_USER"} =$baseauthuser if ($authuser ne $baseauthuser);
        $ENV{"REMOTE_PASS"} = $authpass if (defined($authpass) &&
                                        $config{'pass_password'});
        $ENV{"SSL_USER"} = $peername if ($validated == 2);
        $ENV{"ANONYMOUS_USER"} = "1" if ($validated == 3);
        $ENV{"DOCUMENT_ROOT"} = $config{"root"};
        $ENV{"GATEWAY_INTERFACE"} = "CGI/1.1";
        $ENV{"SERVER_PROTOCOL"} = "HTTP/1.0";
        $ENV{"REQUEST_METHOD"} = $method;
        $ENV{"SCRIPT_NAME"} = $scriptname;
        $ENV{"SCRIPT_FILENAME"} = $full;
        $ENV{"REQUEST_URI"} = $request_uri;
        $ENV{"PATH_INFO"} = $pathinfo;
        $ENV{"PATH_TRANSLATED"} = "$config{root}/$pathinfo";
        $ENV{"QUERY_STRING"} = $querystring;
        $ENV{"MINISERV_CONFIG"} = $conf;
        $ENV{"HTTPS"} = "ON" if ($use_ssl || $config{'inetd_ssl'});
        $ENV{"SESSION_ID"} = $session_id if ($session_id);
        $ENV{"LOCAL_USER"} = $localauth_user if ($localauth_user);
        $ENV{"MINISERV_INTERNAL"} = $miniserv_internal if ($miniserv_internal);
        if (defined($header{"content-length"})) {
                $ENV{"CONTENT_LENGTH"} = $header{"content-length"};
                }
        if (defined($header{"content-type"})) {
```

```
		$ENV{"CONTENT_TYPE"} = $header{"content-type"};
		}
	foreach $h (keys %header) {
		($hname = $h) =~ tr/a-z/A-Z/;
		$hname =~ s/\-/_/g;
		$ENV{"HTTP_$hname"} = $header{$h};
		}
	$ENV{"PWD"} = $cgi_pwd;
	foreach $k (keys %config) {
		if ($k =~ /^env_(\S+)$/) {
			$ENV{$1} = $config{$k};
			}
		}
	delete($ENV{'HTTP_AUTHORIZATION'});
	$ENV{'HTTP_COOKIE'} =~ s/;?\s*$sidname=([a-f0-9]+)//;

	# Check if the CGI can be handled internally
	open(CGI, $full);
	local $first = <CGI>;
	close(CGI);
	$first =~ s/[#!\r\n]//g;
	$nph_script = ($full =~ /\/nph-([^\/]+)$/);
	seek(STDERR, 0, 2);
	if (!$config{'forkcgis'} && $first eq $perl_path && $] >= 5.004) {
		# setup environment for eval
		chdir($ENV{"PWD"});
		@ARGV = split(/\s+/, $queryargs);
		$0 = $full;
		if ($posted_data) {
			# Already read the post input
			$postinput = $posted_data;
			}
		$clen = $header{"content-length"};
		if ($method eq "POST" && $clen_read < $clen) {
			# Still some more POST data to read
			while(length($postinput) < $clen) {
				$buf = &read_data($clen - length($postinput));
				if (!length($buf)) {
					&http_error(500, "Failed to read ".
							"POST request");
					}
				$postinput .= $buf;
				}
			}
		$SIG{'CHLD'} = 'DEFAULT';
		eval {
			# Have SOCK closed if the perl exec's something
			use Fcntl;
			fcntl(SOCK, F_SETFD, FD_CLOEXEC);
			};
		shutdown(SOCK, 0);

		if ($config{'log'}) {
```

```
                        open(MINISERVLOG, ">>$config{'logfile'}");
                        chmod(0600, $config{'logfile'});
                        }
                $doing_eval = 1;
                $main_process_id = $$;
                eval {
                        package main;
                        tie(*STDOUT, 'miniserv');
                        tie(*STDIN, 'miniserv');
                        do $miniserv::full;
                        die $@ if ($@);
                        };
                $doing_eval = 0;
                if ($@) {
                        # Error in perl!
                        &http_error(500, "Perl execution failed", $@);
                        }
                elsif (!$doneheaders && !$nph_script) {
                        &http_error(500, "Missing Headers");
                        }
                #close(SOCK);
                $rv = 0;
                }
        else {
                # fork the process that actually executes the CGI
                pipe(CGIINr, CGIINw);
                pipe(CGIOUTr, CGIOUTw);
                pipe(CGIERRr, CGIERRw);
                if (!($cgipid = fork())) {
                        chdir($ENV{"PWD"});
                        close(SOCK);
                        open(STDIN, "<&CGIINr");
                        open(STDOUT, ">&CGIOUTw");
                        open(STDERR, ">&CGIERRw");
                        close(CGIINw); close(CGIOUTr); close(CGIERRr);
                        exec($full, split(/\s+/, $queryargs));
                        print STDERR "Failed to exec $full : $!\n";
                        exit;
                        }
                close(CGIINr); close(CGIOUTw); close(CGIERRw);

                # send post data
                if ($posted_data) {
                        # already read the posted data
                        print CGIINw $posted_data;
                        }
                $clen = $header{"content-length"};
                if ($method eq "POST" && $clen_read < $clen) {
                        $got = $clen_read;
                        while($got < $clen) {
                                $buf = &read_data($clen-$got);
                                if (!length($buf)) {
                                        kill('TERM', $cgipid);
                                        &http_error(500, "Failed to read ".
                                                      "POST request");
                                        }
                                $got += length($buf);
                                print CGIINw $buf;
                                }
                }
        close(CGIINw);
        shutdown(SOCK, 0);

        if (!$nph_script) {
                # read back cgi headers
                select(CGIOUTr); $|=1; select(STDOUT);
                $got_blank = 0;
                while(1) {
                        $line = <CGIOUTr>;
                        $line =~ s/\r|\n//g;
                        if ($line eq "") {
                                if ($got_blank || %cgiheader) { last; }
                                $got_blank++;
                                next;
                                }
                        ($line =~ /^(\S+):\s+(.*)$/) ||
                                &http_error(500, "Bad Header",
                                            &read_errors(CGIERRr));
                        $cgiheader{lc($1)} = $2;
                        push(@cgiheader, [ $1, $2 ]);
                        }
                if ($cgiheader{"location"}) {
                        &write_data("HTTP/1.0 302 Moved Temporarily\r\n");
                        &write_data("Date: $datestr\r\n");
                        &write_data("Server: $config{'server'}\r\n");
                        &write_keep_alive(0);
                        # ignore the rest of the output. This is a hack, but
                        # is necessary for IE in some cases :(
                        close(CGIOUTr); close(CGIERRr);
                        }
                elsif ($cgiheader{"content-type"} eq "") {
                        &http_error(500, "Missing Content-Type Header",
                                &read_errors(CGIERRr));
                        }
                else {
                        &write_data("HTTP/1.0 $ok_code $ok_message\r\n");
                        &write_data("Date: $datestr\r\n");
                        &write_data("Server: $config{'server'}\r\n");
                        &write_keep_alive(0);
                        }
                foreach $h (@cgiheader) {
                        &write_data("$h->[0]: $h->[1]\r\n");
                        }
                &write_data("\r\n");
                }
        &reset_byte_count();
```

```
                while($line = <CGIOUTr>) {
                        &write_data($line);
                        }
                close(CGIOUTr); close(CGIERRr);
                $rv = 0;
                }
        }
else {
        # A file to output
        open(FILE, $full) || &http_error(404, "Failed to open file");
        &write_data("HTTP/1.0 $ok_code $ok_message\r\n");
        &write_data("Date: $datestr\r\n");
        &write_data("Server: $config{server}\r\n");
        &write_data("Content-type: ".&get_type($full)."\r\n");
        &write_data("Content-length: $stfull[7]\r\n");
        &write_data("Last-Modified: ".&http_date($stfull[9])."\r\n");
        $rv = &write_keep_alive();
        &write_data("\r\n");
        &reset_byte_count();
        while(read(FILE, $buf, 1024) > 0) {
                &write_data($buf);
                }
        close(FILE);
        }

# log the request
&log_request($acpthost, $authuser, $reqline,
             $logged_code ? $logged_code :
             $cgiheader{"location"} ? "302" : $ok_code, &byte_count());
return $rv;
}

# http_error(code, message, body, [dontexit])
sub http_error
{
local $eh = $error_handler_recurse ? undef :
            $config{"error_handler_$_[0]"} ? $config{"error_handler_$_[0]"} :
            $config{'error_handler'} ? $config{'error_handler'} : undef;
if ($eh) {
        # Call a CGI program for the error
        $page = "/$eh";
        $querystring = "code=$_[0]&message=".&urlize($_[1]).
                       "&body=".&urlize($_[2]);
        $error_handler_recurse++;
        $ok_code = $_[0];
        $ok_message = $_[1];
        goto rerun;
        }
else {
        # Use the standard error message display
        &write_data("HTTP/1.0 $_[0] $_[1]\r\n");
        &write_data("Server: $config{server}\r\n");
        &write_data("Date: $datestr\r\n");
        &write_data("Content-type: text/html\r\n");
        &write_keep_alive(0);
        &write_data("\r\n");
        &reset_byte_count();
        &write_data("<h1>Error - $_[1]</h1>\n");
        if ($_[2]) {
                &write_data("<pre>$_[2]</pre>\n");
                }
        }
&log_request($acpthost, $authuser, $reqline, $_[0], &byte_count())
        if ($reqline);
&log_error($_[1], $_[2] ? " : $_[2]" : "");
shutdown(SOCK, 1);
exit if (!$_[3]);
}

sub get_type
{
if ($_[0] =~ /\.([A-z0-9]+)$/) {
        $t = $mime{$1};
        if ($t ne "") {
                return $t;
                }
        }
return "text/plain";
}

# simplify_path(path, bogus)
# Given a path, maybe containing stuff like ".." and "." convert it to a
# clean, absolute form.
sub simplify_path
{
local($dir, @bits, @fixedbits, $b);
$dir = $_[0];
$dir =~ s/^\/+//g;
$dir =~ s/\/+$//g;
@bits = split(/\/+/, $dir);
@fixedbits = ();
$_[1] = 0;
foreach $b (@bits) {
    if ($b eq ".") {
        # Do nothing..
        }
    elsif ($b eq "..") {
        # Remove last dir
        if (scalar(@fixedbits) == 0) {
            $_[1] = 1;
            return "/";
            }
        pop(@fixedbits);
        }
    else {
        # Add dir to list
```

```
            push(@fixedbits, $b);
            }
      }
return "/" . join('/', @fixedbits);
}

# b64decode(string)
# Converts a string from base64 format to normal
sub b64decode
{
   local($str) = $_[0];
   local($res);
   $str =~ tr|A-Za-z0-9+=/||cd;
   $str =~ s/=+$//;
   $str =~ tr|A-Za-z0-9+/| -_|;
   while ($str =~ /(.{1,60})/gs) {
      my $len = chr(32 + length($1)*3/4);
      $res .= unpack("u", $len . $1 );
   }
   return $res;
}

# ip_match(remoteip, localip, [match]+)
# Checks an IP address against a list of IPs, networks and networks/masks
sub ip_match
{
local(@io, @mo, @ms, $i, $j, $hn, $needhn);
@io = split(/\./, $_[0]);
for($i=2; $i<@_; $i++) {
          $needhn++ if ($_[$i] =~ /^\*(\S+)$/);
          }
if ($needhn && !defined($hn = $ip_match_cache{$_[0]})) {
          $hn = gethostbyaddr(inet_aton($_[0]), AF_INET);
          $hn = "" if (&to_ipaddress($hn) ne $_[0]);
          $ip_match_cache{$_[0]} = $hn;
          }
for($i=2; $i<@_; $i++) {
          local $mismatch = 0;
          if ($_[$i] =~ /^(\S+)\/(\S+)$/) {
                    # Compare with network/mask
                    @mo = split(/\./, $1); @ms = split(/\./, $2);
                    for($j=0; $j<4; $j++) {
                              if ((int($io[$j]) & int($ms[$j])) != int($mo[$j])) {
                                        $mismatch = 1;
                                        }
                              }
                    }
          elsif ($_[$i] =~ /^\*(\S+)$/) {
                    # Compare with hostname regexp
                    $mismatch = 1 if ($hn !~ /$1$/);
                    }
          elsif ($_[$i] eq 'LOCAL') {
                    # Compare with local network
```

```
				local @lo = split(/\./, $_[1]);
				if ($lo[0] < 128) {
					$mismatch = 1 if ($lo[0] != $io[0]);
					}
				elsif ($lo[0] < 192) {
					$mismatch = 1 if ($lo[0] != $io[0] ||
							$lo[1] != $io[1]);
					}
				else {
					$mismatch = 1 if ($lo[0] != $io[0] ||
							$lo[1] != $io[1] ||
							$lo[2] != $io[2]);
					}
				}
		elsif ($_[$i] !~ /^[0-9\.]+$/) {
				# Compare with hostname
				$mismatch = 1 if ($_[0] ne &to_ipaddress($_[$i]));
				}
		else {
				# Compare with IP or network
				@mo = split(/\./, $_[$i]);
				while(@mo && !$mo[$#mo]) { pop(@mo); }
				for($j=0; $j<@mo; $j++) {
					if ($mo[$j] != $io[$j]) {
						$mismatch = 1;
						}
					}
				}
		return 1 if (!$mismatch);
		}
return 0;
}

# users_match(&uinfo, user, ...)
# Returns 1 if a user is in a list of users and groups
sub users_match
{
local $uinfo = shift(@_);
local $u;
local @ginfo = getgrgid($uinfo->[3]);
foreach $u (@_) {
		if ($u =~ /^\@(\S+)$/) {
				local @ginfo = getgrnam($1);
				return 1 if ($ginfo[2] == $uinfo->[3]);
				local $m;
				foreach $m (split(/\s+/, $ginfo[3])) {
					return 1 if ($m eq $uinfo->[0]);
					}
				}
		elsif ($u =~ /^(\d*)-(\d*)$/ && ($1 || $2)) {
				return (!$1 || $uinfo[2] >= $1) &&
				       (!$2 || $uinfo[2] <= $2);
				}
```

```
        else {
                return 1 if ($u eq $uinfo->[0]);
                }
        }
return 0;
}

# restart_miniserv()
# Called when a SIGHUP is received to restart the web server. This is done
# by exec()ing perl with the same command line as was originally used
sub restart_miniserv
{
close(SOCK);
&close_all_sockets();
foreach $p (@passin) { close($p); }
foreach $p (@passout) { close($p); }
kill('KILL', $logclearer) if ($logclearer);
kill('KILL', $extauth) if ($extauth);
exec($perl_path, $miniserv_path, @miniserv_argv);
die "Failed to restart miniserv with $perl_path $miniserv_path";
}

sub trigger_restart
{
$need_restart = 1;
}

sub to_ipaddress
{
local (@rv, $i);
foreach $i (@_) {
        if ($i =~ /(\S+)\/(\S+)/ || $i =~ /^\*\S+$/ ||
          $i eq 'LOCAL' || $i =~ /^[0-9\.]+$/) { push(@rv, $i); }
        else { push(@rv, join('.', unpack("CCCC", inet_aton($i)))); }
        }
return wantarray ? @rv : $rv[0];
}

# read_line()
# Reads one line from SOCK or SSL
sub read_line
{
local($idx, $more, $rv);
if (!$use_ssl) {
        # Read a character at a time
    while(1) {
        local $buf;
        local $ok = read(SOCK, $buf, 1);
        if ($ok <= 0) {
            return $rv;
            }
        $rv .= $buf;
        if ($buf eq "\n") {
```

```
                        return $rv;
                        }
                }
                }
while(($idx = index($read_buffer, "\n")) < 0) {
                if (length($read_buffer) > 10000) {
                        &http_error(414, "Request too long",
                                "Received excessive line <pre>$read_buffer</pre>");
                        }

                # need to read more..
                $more = Net::SSLeay::read($ssl_con);
                if ($more eq '') {
                        # end of the data
                        $rv = $read_buffer;
                        undef($read_buffer);
                        return $rv;
                        }
                $read_buffer .= $more;
                }
$rv = substr($read_buffer, 0, $idx+1);
$read_buffer = substr($read_buffer, $idx+1);
return $rv;
}

# read_data(length)
# Reads up to some amount of data from SOCK or the SSL connection
sub read_data
{
local ($rv);
if (length($read_buffer)) {
                $rv = $read_buffer;
                undef($read_buffer);
                return $rv;
                }
elsif ($use_ssl) {
                return Net::SSLeay::read($ssl_con, $_[0]);
                }
else {
                local $buf;
                read(SOCK, $buf, $_[0]) || return undef;
                return $buf;
                }
}

# write_data(data)
# Writes a string to SOCK or the SSL connection
sub write_data
{
if ($use_ssl) {
                Net::SSLeay::write($ssl_con, $_[0]);
                }
else {
		syswrite(SOCK, $_[0], length($_[0]));
		}
$write_data_count += length($_[0]);
}

# reset_byte_count()
sub reset_byte_count { $write_data_count = 0; }

# byte_count()
sub byte_count { return $write_data_count; }

# log_request(hostname, user, request, code, bytes)
sub log_request
{
if ($config{'log'}) {
		local ($user, $ident, $headers);
		if ($config{'logident'}) {
			# add support for rfc1413 identity checking here
			}
		else { $ident = "-"; }
		$user = $_[1] ? $_[1] : "-";
		local $dstr = &make_datestr();
		if (fileno(MINISERVLOG)) {
			seek(MINISERVLOG, 0, 2);
			}
		else {
			open(MINISERVLOG, ">>$config{'logfile'}");
			chmod(0600, $config{'logfile'});
			}
		if (defined($config{'logheaders'})) {
			foreach $h (split(/\s+/, $config{'logheaders'})) {
				$headers .= " $h=\"$header{$h}\"";
				}
			}
		elsif ($config{'logclf'}) {
			$headers = " \"$header{'referer'}\" \"$header{'user-agent'}\"";
			}
		else {
			$headers = "";
			}
		print MINISERVLOG "$_[0] $ident $user [$dstr] \"$_[2]\" ",
				  "$_[3] $_[4]$headers\n";
		close(MINISERVLOG);
		}
}

# make_datestr()
sub make_datestr
{
local @tm = localtime(time());
return sprintf "%2.2d/%s/%4.4d:%2.2d:%2.2d:%2.2d %s",
			$tm[3], $make_date_marr[$tm[4]], $tm[5]+1900,
		$tm[2], $tm[1], $tm[0], $timezone;
```

```
}

# log_error(message)
sub log_error
{
seek(STDERR, 0, 2);
print STDERR "[",&make_datestr(),"] ",
        $acpthost ? ( "[",$acpthost,"] " ) : (),
        $page ? ( $page," : " ) : (),
        @_,"\n";
}

# read_errors(handle)
# Read and return all input from some filehandle
sub read_errors
{
local($fh, $_, $rv);
$fh = $_[0];
while(<$fh>) { $rv .= $_; }
return $rv;
}

sub write_keep_alive
{
local $mode;
if ($config{'nokeepalive'}) {
        # Keep alives have been disabled in config
        $mode = 0;
        }
elsif (@childpids > $config{'maxconns'}*.8) {
        # Disable because nearing process limit
        $mode = 0;
        }
elsif (@_) {
        # Keep alive specified by caller
        $mode = $_[0];
        }
else {
        # Keep alive determined by browser
        $mode = $header{'connection'} =~ /keep-alive/i;
        }
&write_data("Connection: ".($mode ? "Keep-Alive" : "close")."\r\n");
return $mode;
}

sub term_handler
{
kill('TERM', @childpids) if (@childpids);
kill('KILL', $logclearer) if ($logclearer);
kill('KILL', $extauth) if ($extauth);
exit(1);
}
```

```
sub http_date
{
local @tm = gmtime($_[0]);
return sprintf "%s, %d %s %d %2.2d:%2.2d:%2.2d GMT",
                    $weekday[$tm[6]], $tm[3], $month[$tm[4]], $tm[5]+1900,
                    $tm[2], $tm[1], $tm[0];
}

sub TIEHANDLE
{
my $i; bless \$i, shift;
}

sub WRITE
{
$r = shift;
my($buf,$len,$offset) = @_;
&write_to_sock(substr($buf, $offset, $len));
}

sub PRINT
{
$r = shift;
$$r++;
&write_to_sock(@_);
}

sub PRINTF
{
shift;
my $fmt = shift;
&write_to_sock(sprintf $fmt, @_);
}

sub READ
{
$r = shift;
substr($_[0], $_[2], $_[1]) = substr($postinput, $postpos, $_[1]);
$postpos += $_[1];
}

sub OPEN
{
#print STDERR "open() called - should never happen!\n";
}

sub READLINE
{
if ($postpos >= length($postinput)) {
          return undef;
          }
local $idx = index($postinput, "\n", $postpos);
if ($idx < 0) {
```

```
          local $rv = substr($postinput, $postpos);
          $postpos = length($postinput);
          return $rv;
          }
else {
          local $rv = substr($postinput, $postpos, $idx-$postpos+1);
          $postpos = $idx+1;
          return $rv;
          }
}

sub GETC
{
return $postpos >= length($postinput) ? undef
                                        : substr($postinput, $postpos++, 1);
}

sub FILENO
{
return fileno(SOCK);
}

sub CLOSE { }

sub DESTROY { }

# write_to_sock(data, ...)
sub write_to_sock
{
local $d;
foreach $d (@_) {
          if ($doneheaders || $miniserv::nph_script) {
                    &write_data($d);
                    }
          else {
                    $headers .= $d;
                    while(!$doneheaders && $headers =~ s/^([^\r\n]*)(\r)?\n//) {
                              if ($1 =~ /^(\S+):\s+(.*)$/) {
                                        $cgiheader{lc($1)} = $2;
                                        push(@cgiheader, [ $1, $2 ]);
                                        }
                              elsif ($1 !~ /\S/) {
                                        $doneheaders++;
                                        }
                              else {
                                        &http_error(500, "Bad Header");
                                        }
                              }
                    if ($doneheaders) {
                              if ($cgiheader{"location"}) {
                                        &write_data(
                                                  "HTTP/1.0 302 Moved Temporarily\r\n");
                                        &write_data("Date: $datestr\r\n");
```

```
						&write_data("Server: $config{server}\r\n");
						&write_keep_alive(0);
						}
					elsif ($cgiheader{"content-type"} eq "") {
						&http_error(500, "Missing Content-Type Header");
						}
					else {
						&write_data("HTTP/1.0 $ok_code $ok_message\r\n");
						&write_data("Date: $datestr\r\n");
						&write_data("Server: $config{server}\r\n");
						&write_keep_alive(0);
						}
					foreach $h (@cgiheader) {
						&write_data("$h->[0]: $h->[1]\r\n");
						}
					&write_data("\r\n");
					&reset_byte_count();
					&write_data($headers);
					}
			}
		}
}

sub verify_client
{
local $cert = Net::SSLeay::X509_STORE_CTX_get_current_cert($_[1]);
if ($cert) {
		local $errnum = Net::SSLeay::X509_STORE_CTX_get_error($_[1]);
		$verified_client = 1 if (!$errnum);
		}
return 1;
}

sub END
{
if ($doing_eval && $$ == $main_process_id) {
		# A CGI program called exit! This is a horrible hack to
		# finish up before really exiting
		shutdown(SOCK, 1);
		close(SOCK);
		close($PASSINw); close($PASSOUTw);
		&log_request($acpthost, $authuser, $reqline,
				$cgiheader{"location"} ? "302" : $ok_code, &byte_count());
		}
}

# urlize
# Convert a string to a form ok for putting in a URL
sub urlize {
  local($tmp, $tmp2, $c);
  $tmp = $_[0];
  $tmp2 = "";
  while(($c = chop($tmp)) ne "") {
        if ($c !~ /[A-z0-9]/) {
                $c = sprintf("%%%2.2X", ord($c));
                }
        $tmp2 = $c . $tmp2;
        }
 return $tmp2;
}

# validate_user(username, password, [noappend])
sub validate_user
{
return ( ) if (!$_[0]);
if (!$users{$_[0]}) {
        # See if this user exists in Unix and can be validated by the same
        # method as the unixauth webmin user
        return ( undef, 0, 1 ) if (!$config{'unixauth'});
        local $up = $users{$config{'unixauth'}};
        return ( undef, 0, 1 ) if (!defined($up));
        local @uinfo = getpwnam($_[0]);

        # Work out our domain name from the hostname
        local $dom = $host;
        if ($dom =~ /^(www|ftp|mail)\.(\S+)$/i) {
                $dom = $2;
                }

        if ($config{'user_mapping'} && !defined(%user_mapping)) {
                # Read the user mapping file
                %user_mapping = ();
                open(MAPPING, $config{'user_mapping'});
                while(<MAPPING>) {
                        s/\r|\n//g;
                        s/#.*$//;
                        if (/^(\S+)\s+(\S+)/) {
                                $user_mapping{$2} = $1;
                                }
                        }
                close(MAPPING);
                }

        # Check the user mapping file to see if there is an entry for the
        # user login in which specifies a new effective user
        local $um = $user_mapping{"$_[0]\@$host"} ||
                    $user_mapping{"$_[0]\@$dom"} ||
                    $user_mapping{$_[0]};
        if (defined($um) && ($_[2]&4) == 0) {
                # A mapping exists - use it!
                local @vu = &validate_user($um, $_[1], $_[2]+4);
                return @vu;
                }

        # Check if a user with the entered login and the domain appended
        # or prepended exists, and if so take it to be the effective user
```

```
if (!@uinfo && $config{'domainuser'}) {
        # Try again with name.domain and name.firstpart
        local $first;
        if ($dom =~ /^([^\.]+)/) {
                $first = $1;
                }
        if (($_[2]&1) == 0) {
                local ($a, $p);
                foreach $a ($first, $dom) {
                        foreach $p ("$_[0].${a}", "$_[0]-${a}",
                                    "${a}.$_[0]", "${a}-$_[0]",
                                    "$_[0]_${a}", "${a}_$_[0]") {
                                local @vu = &validate_user($p, $_[1],
                                                           $_[2] + 1);
                                return @vu if (@vu);
                                }
                        }
                }
        }

# Check if the user entered a domain at the end of his username when
# he really shouldn't have, and if so try without it
if (!@uinfo && $config{'domainstrip'} &&
   $_[0] =~ /^(\S+)\@/ && ($_[2]&2) == 0) {
        local $stripped = $1;
        local @vu = &validate_user($stripped, $_[1], $_[2] + 2);
        return @vu if (@vu);
        }

return ( undef, 0, 1 ) if (!@uinfo);

if (defined(@allowusers)) {
        # Only allow people on the allow list
        return ( undef, 0, 0 ) if (!&users_match(\@uinfo, @allowusers));
        }
elsif (defined(@denyusers)) {
        # Disallow people on the deny list
        return ( undef, 0, 0 ) if (&users_match(\@uinfo, @denyusers));
        }
if ($config{'shells_deny'}) {
        local $found = 0;
        open(SHELLS, $config{'shells_deny'});
        while(<SHELLS>) {
                s/\r|\n//g;
                s/#.*$//;
                $found++ if ($_ eq $uinfo[8]);
                }
        close(SHELLS);
        return ( undef, 0, 0 ) if (!$found);
        }

if ($up eq 'x') {
        local $val = &validate_unix_user($_[0], $_[1]);
```

```
                        return $val == 2 ? ( $_[0], 1, 0 ) :
                            $val == 1 ? ( $_[0], 0, 0 ) : ( undef, 0, 0 );
                        }
                elsif ($up eq 'e') {
                        return &validate_external_user($_[0], $_[1]) ? ( $_[0] ) : ( );
                        }
                else {
                        return $up eq crypt($_[1], $up) ? ( $_[0] ) : ( );
                        }
                }
elsif ($users{$_[0]} eq 'x') {
                # Call PAM to validate the user
                local $val = &validate_unix_user($_[0], $_[1]);
                return $val == 2 ? ( $_[0], 1, 0 ) :
                    $val == 1 ? ( $_[0], 0, 0 ) : ( undef, 0, 0 );
                }
elsif ($users{$_[0]} eq 'e') {
                # Pass to the external authentication program
                return &validate_external_user($_[0], $_[1]) ?
                        ( $_[0], 0, 0 ) : ( undef, 0, 0 );
                }
else {
                # Check against the webmin user list
                return $users{$_[0]} eq crypt($_[1], $users{$_[0]}) ?
                        ( $_[0], 0, 0 ) : ( undef, 0, 0 );
                }
}

# validate_unix_user(user, password)
# Returns 1 if a username and password are valid under unix, 0 if not.
# Checks PAM if available, and falls back to reading the system password
# file otherwise.
sub validate_unix_user
{
if ($use_pam) {
                # Check with PAM
                $pam_username = $_[0];
                $pam_password = $_[1];
                local $pamh = new Authen::PAM($config{'pam'}, $pam_username,
                                              \&pam_conv_func);
                if (ref($pamh)) {
                        local $pam_ret = $pamh->pam_authenticate();
                        return 1 if ($pam_ret == PAM_SUCCESS);
                        }
                }
elsif ($config{'passwd_file'}) {
                local $rv = 0;
                open(FILE, $config{'passwd_file'});
                if ($config{'passwd_file'} eq '/etc/security/passwd') {
                        # Assume in AIX format
                        while(<FILE>) {
                                s/\s*$//;
                                if (/^\s*(\S+):/ && $1 eq $_[0]) {
                                        $_ = <FILE>;
                                        if (/^\s*password\s*=\s*(\S+)\s*$/) {
                                                $rv = $1 eq crypt($_[1], $1) ? 1 : 0;
                                                }
                                        last;
                                        }
                                }
                        }
        else {
                # Read the system password or shadow file
                while(<FILE>) {
                        local @l = split(/:/, $_, -1);
                        local $u = $l[$config{'passwd_uindex'}];
                        local $p = $l[$config{'passwd_pindex'}];
                        if ($u eq $_[0]) {
                                $rv = $p eq crypt($_[1], $p) ? 1 : 0;
                                if ($config{'passwd_cindex'} ne '' && $rv) {
                                        # Password may have expired!
                                        local $c = $l[$config{'passwd_cindex'}];
                                        local $m = $l[$config{'passwd_mindex'}];
                                        local $day = time()/(24*60*60);
                                        if ($c =~ /^\d+/ && $m =~ /^\d+/ &&
                                           $day - $c > $m) {
                                                # Yep, it has ..
                                                $rv = 2;
                                                }
                                        }
                                last;
                                }
                        }
                }
        close(FILE);
        return $rv if ($rv);
        }

# Fallback option - check password returned by getpw*
local @uinfo = getpwnam($_[0]);
if ($uinfo[1] ne '' && crypt($_[1], $uinfo[1]) eq $uinfo[1]) {
        return 1;
        }

return 0; # Totally failed
}

# validate_external_user(user, pass)
# Validate a user by passing the username and password to an external
# squid-style authentication program
sub validate_external_user
{
return 0 if (!$config{'extauth'});
flock(EXTAUTH, 2);
local $str = "$_[0] $_[1]\n";
syswrite(EXTAUTH, $str, length($str));
```

```
local $resp = <EXTAUTH>;
flock(EXTAUTH, 8);
return $resp =~ /^OK/i ? 1 : 0;
}

# the PAM conversation function for interactive logins
sub pam_conv_func
{
$pam_conv_func_called++;
my @res;
while ( @_ ) {
        my $code = shift;
        my $msg = shift;
        my $ans = "";

        $ans = $pam_username if ($code == PAM_PROMPT_ECHO_ON() );
        $ans = $pam_password if ($code == PAM_PROMPT_ECHO_OFF() );

        push @res, PAM_SUCCESS();
        push @res, $ans;
        }
push @res, PAM_SUCCESS();
return @res;
}

sub urandom_timeout
{
close(RANDOM);
}

# get_socket_name(handle)
sub get_socket_name
{
return $config{'host'} if ($config{'host'});
local $sn = getsockname($_[0]);
return undef if (!$sn);
local ($myport, $myaddr) = unpack_sockaddr_in($sn);
if (!$get_socket_name_cache{$myaddr}) {
        local $myname = gethostbyaddr($myaddr, AF_INET);
        if ($myname eq "") {
                $myname = inet_ntoa($myaddr);
                }
        $get_socket_name_cache{$myaddr} = $myname;
        }
return $get_socket_name_cache{$myaddr};
}

# run_login_script(username, sid, remoteip, localip)
sub run_login_script
{
if ($config{'login_script'}) {
        system($config{'login_script'}.
            " ".join(" ", map { quotemeta($_) } @_).
```

```
                " >/dev/null 2>&1 </dev/null");
        }
}

# run_logout_script(username, sid, remoteip, localip)
sub run_logout_script
{
if ($config{'logout_script'}) {
        system($config{'logout_script'}.
                " ".join(" ", map { quotemeta($_) } @_).
                " >/dev/null 2>&1 </dev/null");
        }
}

# close_all_sockets()
# Closes all the main listening sockets
sub close_all_sockets
{
local $s;
foreach $s (@socketfhs) {
        close($s);
        }
}
```

Showing Alerts.pl:

```
#!/usr/bin/perl

print "Content-type: text/html\n";

use CGI;
use DBI;
use strict;

##########################################################################################
# form variables handling


my $CGI=CGI->new();
my $USER=$CGI->param('user');
my $PASS=$CGI->param('pw');
my $HOST=$CGI->param('host');
my $DBNAME=$CGI->param('db');
my $LIMIT=$CGI->param('limit');
my $MINDATE=$CGI->param('mindate');
my $MAXDATE=$CGI->param('maxdate');
my $TRI=$CGI->param('tri');
my $MEMPW=$CGI->param('mempw');
my $TYPE_DEL=$CGI->param('type_del');
my $TARGET_DEL=$CGI->param('target_del');
my $HACKER=$CGI->param('hacker');




# Please feel free to change the Priorities of the groups.
# These groups are taken from Snort.conf

my ($BESTOF_QUERY,$SENSORS_QUERY,$TAB_QUERY,$HUMAN_DATE,%class,@hosts);

$class{'icmp-event'} = "low";
$class{'misc-activity'} = "low";
$class{'network-scan'} = "low";
$class{'not-suspicious'} = "low";
$class{'protocol-command-decode'} = "low";
$class{'string-detect'} = "low";
$class{'unknown'} = "low";

$class{'attempted-dos'} = "medium";
$class{'attempted-recon'} = "medium";
$class{'bad-unknown'} = "medium";
$class{'denial-of-service'} = "medium";
$class{'misc-attack'} = "medium";
$class{'non-standard-protocol'} = "medium";
$class{'rpc-portmap-decode'} = "medium";
$class{'successful-dos'} = "medium";
$class{'successful-recon-largescale'} = "medium";
$class{'successful-recon-limited'} = "medium";
$class{'suspicious-filename-detect'} = "medium";
$class{'suspicious-login'} = "medium";
$class{'system-call-detect'} = "medium";
$class{'unusual-client-port-connection'} = "medium";
$class{'web-application-activity'} = "medium";

$class{'attempted-admin'} = "high";
$class{'attempted-user'} = "high";
$class{'shellcode-detect'} = "high";
$class{'successful-admin'} = "high";
$class{'successful-user'} = "high";
$class{'trojan-activity'} = "high";
$class{'unsuccessful-user'} = "high";
$class{'web-application-attack'} = "high";


#####################################################################################
# Main program


if ($TRI eq ""){
        $TRI = 3;
}
```

```
$HUMAN_DATE=sapiens(time());
&build_report_query();
&head();
&form();


if ($TYPE_DEL eq "6"){
        &delete_all();
        $TYPE_DEL = "1";
}

if ($TARGET_DEL ne ""){

        if ($TYPE_DEL eq "1"){
                &del_by_source();
        }

        if ($TYPE_DEL eq "2"){
                &del_by_dest();
        }

        if ($TYPE_DEL eq "3"){
                &del_by_date();
        }

        if ($TYPE_DEL eq "4"){
                &del_by_signature();
        }

        if ($TYPE_DEL eq "5"){
                &del_by_sensor();
        }
}

if ($PASS ne ""){

        &report();
}
&foot();

##########################################################################################
# reporting SQL queries


sub build_report_query {

        if ($TRI eq "1"){
                $TAB_QUERY="
                SELECT
event.cid,event.sid,inet_ntoa(iphdr.ip_src),inet_ntoa(iphdr.ip_dst),event.timestamp,sig_class.sig_class_nam
e,signature.sig_sid
                FROM iphdr,event,signature,sig_class
                WHERE inet_ntoa(iphdr.ip_src)=\"$HACKER\" and event.signature=signature.sig_id
and iphdr.cid=event.cid and sig_class.sig_class_id=signature.sig_class_id and left(event.timestamp, 10) >=
\"$MINDATE\" and left(event.timestamp, 10) <= \"$MAXDATE\"
                ORDER BY iphdr.ip_src;";
        }
        if ($TRI eq "2"){
                $TAB_QUERY="
                SELECT
event.cid,event.sid,inet_ntoa(iphdr.ip_src),inet_ntoa(iphdr.ip_dst),event.timestamp,sig_class.sig_class_nam
e,signature.sig_sid
                FROM iphdr,event,signature,sig_class
                WHERE inet_ntoa(iphdr.ip_src)=\"$HACKER\" and event.signature=signature.sig_id
and iphdr.cid=event.cid and sig_class.sig_class_id=signature.sig_class_id and left(event.timestamp, 10) >=
\"$MINDATE\" and left(event.timestamp, 10) <= \"$MAXDATE\"
                ORDER BY iphdr.ip_dst;";
        }
        if ($TRI eq "3"){
                $TAB_QUERY="
                SELECT
event.cid,event.sid,inet_ntoa(iphdr.ip_src),inet_ntoa(iphdr.ip_dst),event.timestamp,sig_class.sig_class_nam
e,signature.sig_sid
                FROM iphdr,event,signature,sig_class
                WHERE inet_ntoa(iphdr.ip_src)=\"$HACKER\" and event.signature=signature.sig_id
and iphdr.cid=event.cid and sig_class.sig_class_id=signature.sig_class_id and left(event.timestamp, 10) >=
\"$MINDATE\" and left(event.timestamp, 10) <= \"$MAXDATE\"
                ORDER BY event.timestamp DESC;";
        }
        if ($TRI eq "4"){
                $TAB_QUERY="
                SELECT
event.cid,event.sid,inet_ntoa(iphdr.ip_src),inet_ntoa(iphdr.ip_dst),event.timestamp,sig_class.sig_class_nam
e,signature.sig_sid
                FROM iphdr,event,signature,sig_class
                WHERE inet_ntoa(iphdr.ip_src)=\"$HACKER\" and event.signature=signature.sig_id
and iphdr.cid=event.cid and sig_class.sig_class_id=signature.sig_class_id and left(event.timestamp, 10) >=
\"$MINDATE\" and left(event.timestamp, 10) <= \"$MAXDATE\"
                ORDER BY event.signature;";
        }
        if ($TRI eq "5"){
                $TAB_QUERY="
                SELECT
event.cid,event.sid,inet_ntoa(iphdr.ip_src),inet_ntoa(iphdr.ip_dst),event.timestamp,sig_class.sig_class_nam
e,signature.sig_sid
                FROM iphdr,event,signature,sig_class
                WHERE inet_ntoa(iphdr.ip_src)=\"$HACKER\" and event.signature=signature.sig_id
and iphdr.cid=event.cid and sig_class.sig_class_id=signature.sig_class_id and left(event.timestamp, 10) >=
\"$MINDATE\" and left(event.timestamp, 10) <= \"$MAXDATE\"
                ORDER BY event.sid;";
        }
        # like \'\%$query\%\';";
}
```

```
###########################################################################################
#  Report generation connecting to MySQL DB using DBI available under GNU license
#  CGI web interface


sub report{
        my ($i,
                $DBH,
                $STH,
                $NUMBEROW,
                $NUMBERFIELDS,
                @COLUMN,
                $REF,
                $tr,
                $SIGID,
                $event_class);


        $DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");


        # <A
HREF=\"header.pl\?db=snort\&user=snort\&host=192.168.1.200\&pw=snortpw\&cid=".$$REF[$i]."\&sid
=".$$REF[$i+1]."\" >$$REF[$i]</a>
        $STH = $DBH->prepare($TAB_QUERY); # sending query to mysql
        $STH->execute();
        $NUMBEROW = $STH->rows;
        print "<H3>$NUMBEROW Alerts(s) from $MINDATE to $MAXDATE :</H3>\n"; # debug
        $NUMBERFIELDS = $STH->{'NUM_OF_FIELDS'};          # number of fields in the result
table
        # $COLUMNAME = $STH->{'NAME'};
        @COLUMN=("ALERT ANALYSIS","SENSOR","SOURCE IP\@","DEST
IP\@","DATE","SIGNATURE","SID","THREAT");

        # printing the result TAB
        print "
        <CENTER>\n
        <TABLE BORDER=0 CELLPADDING=5 bgcolor=\"#aeaaae\" width=\"100\%\">\n
        <TR bgcolor=\"#00008b\">";
        for ($i = 0;  $i < 8;  $i++) {
            print "
            <TD ALIGN=center><B><FONT size=\"1\"
color=\"#ffffff\">@COLUMN[$i]</FONT></B></TD>";
        }
        print "</TR>\n";
        $tr=0;
        while ($REF = $STH->fetchrow_arrayref) {
                $tr++;
                if ($tr % 2) {
                        print "<TR bgcolor=\"#cccccc\">";
                }else{
                        print "<TR bgcolor=\"#bbbbbb\">";
                }
```

```
                    for ($i = 0;  $i < 1;  $i++) {
                            print "
                            <TD ALIGN=center><FONT size=\"1\">
                            <FORM ACTION=\"header.pl\" METHOD= post > \n
                            <INPUT TYPE=\"hidden\" NAME=\"pw\" VALUE=".$PASS.">\n
                            <INPUT TYPE=\"hidden\" NAME=\"host\" VALUE=".$HOST.">\n
                            <INPUT TYPE=\"hidden\" NAME=\"db\" VALUE=".$DBNAME.">\n
                            <INPUT TYPE=\"hidden\" NAME=\"user\" VALUE=".$USER.">\n
                            <INPUT TYPE=\"hidden\" NAME=\"cid\" VALUE=".$$REF[$i].">\n
                            <INPUT TYPE=\"hidden\" NAME=\"sid\" VALUE=".$$REF[$i+1].">\n
                            <INPUT TYPE=\"submit\" NAME=\"submitButtonName\"
VALUE=\"Details\">\n
                            </FORM></FONT></TD>";
                    }
                    for ($i = 1;  $i < 2;  $i++) {
                            print "
                            <TD><CENTER><FONT
size=\"1\">$$REF[$i]</FONT></CENTER></TD>";
                    }
                    for ($i = 2;  $i < 4;  $i++) {
                            print "
                            <TD><FONT size=\"1\">$$REF[$i]</FONT></TD>";
                    }
                    for ($i = 4;  $i < 5;  $i++) {
                            print "
                            <TD><FONT size=\"1\">$$REF[$i]</FONT></TD>";
                    }
                    for ($i = 5;  $i < 6;  $i++) {
                            print "
                            <TD><FONT size=\"1\">$$REF[$i]</FONT></TD>";
                            $event_class=$$REF[$i];
                    }
                    for ($i = 6;  $i < 7;  $i++) {
                            print "
                            <TD><FONT size=\"1\">$$REF[$i]</FONT></TD>";
                            $SIGID=$$REF[$i];
                    }
                    if ($class{$event_class} eq "low"){
                            print "
                            <TD ALIGN=center><A HREF=\"http://www.snort.org/snort-
db/sid.html?sid=".$SIGID."\" TARGET=new>
                            <IMG BORDER=0 width=22 height=22 SRC=\"./safe.gif\"
ALT=\"Whois\"></A></TD>";
                    }
                    elsif ($class{$event_class} eq "medium"){
                            print "
                            <TD ALIGN=center><A HREF=\"http://www.snort.org/snort-
db/sid.html?sid=".$SIGID."\" TARGET=new>
                            <IMG BORDER=0 width=22 height=22 SRC=\"./medium.gif\"
ALT=\"Whois\"></A></TD>";
                    }
                    elsif ($class{$event_class} eq "high"){
```

```
			print "
			<TD ALIGN=center><A HREF=\"http://www.snort.org/snort-
db/sid.html?sid=".$SIGID."\" TARGET=new>
			<IMG BORDER=0 width=22 height=22 SRC=\"./critical.gif\"
ALT=\"Whois\"></A></TD>";
		}
		else{
			print "
			<TD ALIGN=center><A HREF=\"http://www.snort.org/snort-
db/sid.html?sid=".$SIGID."\" TARGET=new>
			<IMG BORDER=0 width=22 height=22 SRC=\"./unknown.gif\"
ALT=\"Whois\"></A></TD>";
		}
		print "</TR>\n";
	}
	print "
	</TABLE></CENTER>\n";
	#print "<P><B>sql :  </B>".$TAB_QUERY."</p>\n";		# debug
	$STH->finish();
	$DBH->disconnect();
}

##########################################################################################
# Delete by destination


sub del_by_dest {
	my (@TABLES,
		$STH,
		$DBH,
		$REF,
		$DEL_QUERY,
		$i,
		$j,
		$DELETE_REQUEST,
		@SID,
		@CID);

	# Searching for common CID		#

	$DEL_QUERY="
		SELECT iphdr.cid
		FROM iphdr
		WHERE iphdr.ip_dst = inet_aton(\"$TARGET_DEL\") and iphdr.ip_src =
inet_aton(\"$HACKER\");";

	$DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");

	$STH = $DBH->prepare($DEL_QUERY); # sending query to MYSQL
	$STH->execute();
	while ($REF = $STH->fetchrow_arrayref) {
		push(@CID,$$REF[$i]);
	}
```

```
        # Deleting          events #

        @TABLES = ('event','iphdr','icmphdr','tcphdr','udphdr','opt','data');
        for ($j = 0;  $j < $#CID+1;  $j++) {
                for ($i = 0;  $i < $#TABLES+1;      $i++) {
                        $DELETE_REQUEST="DELETE FROM @TABLES[$i] WHERE
@TABLES[$i].cid = \"@CID[$j]\";";
                        $STH = $DBH->prepare($DELETE_REQUEST);
                        $STH->execute();
                }
        }
        $STH->finish();
        $DBH->disconnect();
}

##########################################################################################
# Delete by date

sub del_by_date {
        my (@TABLES, $STH,   $DBH, $REF, $DEL_QUERY, $i, $j, $DELETE_REQUEST, @SID,
                @CID);


        # Searching for common CID

        $DEL_QUERY="
                SELECT event.cid
                FROM event
                WHERE event.timestamp = \"$TARGET_DEL\" and iphdr.ip_src =
inet_aton(\"$HACKER\");";

        $DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");

        $STH = $DBH->prepare($DEL_QUERY); # sending query to MYSQL
        $STH->execute();
        while ($REF = $STH->fetchrow_arrayref) {
                push(@CID,$$REF[$i]);
        }


        # Deleting events

        @TABLES = ('event','iphdr','icmphdr','tcphdr','udphdr','opt','data');
        for ($j = 0;  $j < $#CID+1;  $j++) {
                for ($i = 0;  $i < $#TABLES+1;      $i++) {
                        $DELETE_REQUEST="DELETE FROM @TABLES[$i] WHERE
@TABLES[$i].cid = \"@CID[$j]\";";
                        $STH = $DBH->prepare($DELETE_REQUEST);
                        $STH->execute();
                }
        }
        $STH->finish();
        $DBH->disconnect();
}
########################################################################################
# Delete by signature


sub del_by_signature {
        my (@TABLES, $STH, $DBH, $REF, $DEL_QUERY, $i, $j, $DELETE_REQUEST, @SID,
                @CID);


        # Searching for common CID

        $DEL_QUERY="
                SELECT event.cid
                FROM event
                WHERE event.signature = \"$TARGET_DEL\" and iphdr.ip_src =
inet_aton(\"$HACKER\");";

        $DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");

        $STH = $DBH->prepare($DEL_QUERY); # sending query to MYSQL
        $STH->execute();
        while ($REF = $STH->fetchrow_arrayref) {
                push(@CID,$$REF[$i]);
        }

        # Deleting events

        @TABLES = ('event','iphdr','icmphdr','tcphdr','udphdr','opt','data');
        for ($j = 0;  $j < $#CID+1;  $j++) {
                for ($i = 0;  $i < $#TABLES+1;      $i++) {
                        $DELETE_REQUEST="DELETE FROM @TABLES[$i] WHERE
@TABLES[$i].cid = \"@CID[$j]\";";
                        $STH = $DBH->prepare($DELETE_REQUEST);
                        $STH->execute();
                }
        }

        # Deleting signatures

        $DELETE_REQUEST="DELETE FROM signature WHERE signature.sig_sid =
\"$TARGET_DEL\";";
        $STH = $DBH->prepare($DELETE_REQUEST);
        $STH->execute();

        $STH->finish();
        $DBH->disconnect();
}



# Delete by SENSOR
```

```
sub del_by_sensor {
        my (@TABLES,  $STH, $DBH, $REF, $DEL_QUERY, $i, $j, $DELETE_REQUEST, @SID,
                @CID);


        # Searching for common CID

        $DEL_QUERY="
                SELECT event.cid
                FROM event
                WHERE event.sid = \"$TARGET_DEL\" and iphdr.ip_src = inet_aton(\"$HACKER\");";

        $DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");

        $STH = $DBH->prepare($DEL_QUERY); # sending query to MYSQL
        $STH->execute();
        while ($REF = $STH->fetchrow_arrayref) {
                push(@CID,$$REF[$i]);
        }

        # Deleting events


        @TABLES = ('event','iphdr','icmphdr','tcphdr','udphdr','opt','data');
        for ($j = 0;  $j < $#CID+1;  $j++) {
                for ($i = 0;  $i < $#TABLES+1;      $i++) {
                        $DELETE_REQUEST="DELETE FROM @TABLES[$i] WHERE
@TABLES[$i].cid = \"@CID[$j]\";";
                        $STH = $DBH->prepare($DELETE_REQUEST);
                        $STH->execute();
                }
        }


        # Deleting sensors

        $DELETE_REQUEST="DELETE FROM sensor WHERE sensor.sid = \"$TARGET_DEL\";";
        $STH = $DBH->prepare($DELETE_REQUEST);
        $STH->execute();

        $STH->finish();
        $DBH->disconnect();
}
#########################################################################################
# Delete all

sub delete_all {

        my (@TABLES,  $STH, $DBH, $REF, $DEL_QUERY, $i, $j, $DELETE_REQUEST, @SID,
                @CID);
        # Searching for common CID

        $DEL_QUERY="
                SELECT iphdr.cid
                FROM iphdr
                WHERE iphdr.ip_src = inet_aton(\"$HACKER\");";

        $DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");

        $STH = $DBH->prepare($DEL_QUERY); # sending query to MYSQL
        $STH->execute();
        while ($REF = $STH->fetchrow_arrayref) {
                push(@CID,$$REF[$i]);
        }


        # Deleting events

        @TABLES = ('event','iphdr','icmphdr','tcphdr','udphdr','opt','data');
        for ($j = 0;  $j < $#CID+1;  $j++) {
                for ($i = 0;  $i < $#TABLES+1;      $i++) {
                        $DELETE_REQUEST="DELETE FROM @TABLES[$i] WHERE
@TABLES[$i].cid = \"@CID[$j]\";";
                        $STH = $DBH->prepare($DELETE_REQUEST);
                        $STH->execute();
                }
        }
        $STH->finish();
        $DBH->disconnect();
}


###########################################################################################
# FORM

sub form {
        print "
        <TABLE BORDER=0 CELLPADDING=6 bgcolor=\"#00008b\" width=\"100\%\" ><TR> \n
        <TD ALIGN=center><B><FONT color=\"#339966\">REPORT GENERATED on
$HUMAN_DATE </FONT></B></TD>
        <TD ALIGN=center><B><FONT color=\"#ffffff\"></FONT></B></TD> \n
        </TR></TABLE>\n

        <FORM ACTION=\"alerts.pl\" METHOD= post > \n
        <INPUT TYPE=\"hidden\" NAME=\"pw\" VALUE=".$PASS.">\n
        <INPUT TYPE=\"hidden\" NAME=\"host\" VALUE=".$HOST.">\n
        <INPUT TYPE=\"hidden\" NAME=\"db\" VALUE=".$DBNAME.">\n
        <INPUT TYPE=\"hidden\" NAME=\"user\" VALUE=".$USER.">\n
        <INPUT TYPE=\"hidden\" NAME=\"mempw\" VALUE=".$MEMPW.">\n
        <INPUT TYPE=\"hidden\" NAME=\"hacker\" VALUE=".$HACKER.">\n
        <INPUT TYPE=\"hidden\" NAME=\"limit\" VALUE=".$LIMIT.">\n
        <INPUT TYPE=\"hidden\" NAME=\"mindate\" VALUE=".$MINDATE.">\n
        <INPUT TYPE=\"hidden\" NAME=\"maxdate\" VALUE=".$MAXDATE.">\n
        <P><H3>Query options :</H3></P>\n
        <TABLE BORDER=0 CELLPADDING=6 bgcolor=\"#00008b\" width=\"100\%\" ><TR> \n
        <TD ALIGN=center><B><FONT color=\"#ffffff\">SORT ALERTS BY :</FONT></B></TD>
        <TD ALIGN=center><B><FONT color=\"#ffffff\">DELETE FROM DB BY
:</FONT></B></TD> \n
        <TD ALIGN=center rowspan=2> \n
        <P><INPUT TYPE=\"submit\" NAME=\"submitButtonName\" VALUE=\"Send query\"> \n
        <INPUT TYPE=\"reset\" VALUE=\"Reset\"></P> \n
        </TD></TR>\n
        <TR>\n
        <TD> \n
        <CENTER><SELECT NAME=\"tri\"> \n";
#       if ($TRI eq "1"){
#               print "<OPTION VALUE=\"1\" SELECTED>Source \n";
#       }else{
#               print "<OPTION VALUE=\"1\">Source \n";
#       }
        if ($TRI eq "2"){
                print "<OPTION VALUE=\"2\" SELECTED>Destination \n";
        }else{
                print "<OPTION VALUE=\"2\">Destination \n";
        }
        if (($TRI eq "3") || ($TRI eq "")){
                print "<OPTION VALUE=\"3\" SELECTED>Date \n";
        }else{
                print "<OPTION VALUE=\"3\">Date \n";
        }
        if ($TRI eq "4"){
                print "<OPTION VALUE=\"4\" SELECTED>Signature \n";
        }else{
                print "<OPTION VALUE=\"4\">Signature \n";
        }
        if ($TRI eq "5"){
                print "<OPTION VALUE=\"5\" SELECTED>Sensor \n";
        }else{
                print "<OPTION VALUE=\"5\">Sensor \n";
        }
        print "
        </SELECT></CENTER> \n
        </TD> \n
  <TD> \n
        <CENTER><SELECT NAME=\"type_del\"> \n";
#       if (($TYPE_DEL eq "1") || ($TRI eq "")){
#               print "<OPTION VALUE=\"1\" SELECTED>Source \n";
#       }else{
#               print "<OPTION VALUE=\"1\">Source \n";
#       }
        if ($TYPE_DEL eq "2"){
                print "<OPTION VALUE=\"2\" SELECTED>Destination \n";
        }else{
                print "<OPTION VALUE=\"2\">Destination \n";
        }
        if ($TYPE_DEL eq "3"){
                print "<OPTION VALUE=\"3\" SELECTED>Date \n";
        }else{
                print "<OPTION VALUE=\"3\">Date \n";
        }
        if ($TYPE_DEL eq "4"){
                print "<OPTION VALUE=\"4\" SELECTED>Signature(ID) \n";
        }else{
                print "<OPTION VALUE=\"4\">Signature(ID) \n";
        }
        if ($TYPE_DEL eq "5"){
                print "<OPTION VALUE=\"5\" SELECTED>Sensor(ID) \n";
        }else{
                print "<OPTION VALUE=\"5\">Sensor(ID) \n";
        }
        print "<OPTION VALUE=\"6\">All(\! \! \!) \n";
        print "
        </SELECT><FONT color=\"#ffffff\"> = \n</FONT>
        <INPUT TYPE=\"text\" NAME=\"target_del\" size=\"24\"></CENTER>
  </TD></TR></TABLE> \n
  </FORM> \n
  <BR> \n";
}

#########################################################################################
# Header


sub head{
        print "
        <HTML><HEAD>\n
        <BODY lang=FR bgcolor=\"#f4f4e0\" style='FONT-size:08.0pt\;FONT-family:Sans'> \n
        <TABLE ALIGN=center BORDER=0 CELLPADDING=6 bgcolor=\"#f4f4e0\" width=\"100\%\"
><TR> \n
        <TD><CENTER><A HREF=\"http://www.ripe.net/\" TARGET=new>
        <IMG BORDER=0 width=50 height=22 SRC=\"./ripe.gif\"
ALT=\"Whois\"></A></CENTER></TD> \n
        <TD><CENTER><A HREF=\"http://www.nic.fr/zonecheck/\" TARGET=new>
        <IMG BORDER=0 width=50 height=22 SRC=\"./afnic.gif\"
ALT=\"Zonecheck\"></A></CENTER></TD> \n
        <TD><CENTER><A HREF=\"http://www.snort.org/snort-db/\" TARGET=new>
        <IMG BORDER=0 width=50 height=22 SRC=\"./snort.gif\"
ALT=\"SnortDB\"></A></CENTER></TD> \n
        <TD><CENTER><A HREF=\"http://online.securityfocus.com/bid/\" TARGET=new>
        <IMG BORDER=0 width=50 height=22 SRC=\"./focus.gif\"
ALT=\"SecurityFocus\"></A></CENTER></TD> \n
        </TR></TABLE> \n
        <TITLE>SNORT</TITLE>\n
        <A name=top></A> \n
        <H1 ALIGN=center>SNORT Reporter</H1> \n
        </HEAD> \n
        <P ALIGN=center>Reporting and investigation for the SNORT Network Intrusion Detection
System.</P> \n";
```

```
}

#############################################################################################
# Footer

sub foot{
        print "
        <A
HREF=\"http://gaia.ecs.csus.edu/~bhatian/snortreporting/sr.html\">http://gaia.ecs.csus.edu/csc</a><BR>\n
        Report created on : $HUMAN_DATE </P> \n
        <P ALIGN=center><A HREF=\"\#top\">Top of the page</a></P> \n
        </BODY></HTML>";
}

#############################################################################################
# date formating

sub sapiens {
        my
($HUMAN_DATE,$SEC,$MIN,$HOUR,$MDAY,$MONTH,$YEAR,$SDAY,$ADAY,$ISDST);
        ($SEC,$MIN,$HOUR,$MDAY,$MONTH,$YEAR,$SDAY,$ADAY,$ISDST) = localtime($_[0]);
        $YEAR = ($YEAR-100);
        $YEAR ="200$YEAR";
        $MONTH++;
        if ($MONTH < 10){
                $MONTH="0".$MONTH;
        }
        if ($MDAY < 10){
                $MDAY="0".$MDAY;
        }
        if ($HOUR < 10){
                $HOUR="0".$HOUR;
        }
        if ($MIN < 10){
                $MIN="0".$MIN;
        }
        if ($SEC < 10){
                $SEC="0".$SEC;
        }
        #@LIST = ('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec');
        #$MONTH = $LIST[$MONTH];
        $HUMAN_DATE = "$YEAR-$MONTH-$MDAY $HOUR:$MIN:$SEC ";
return $HUMAN_DATE
}
```

Showing Header.pl:

```
#!/usr/bin/perl

print "Content-type: text/html\n";
```

```
# GNU GENERAL PUBLIC LICENSE http://www.gnu.org/copyleft/gpl.html

use CGI;
use DBI;
use strict;
##########################################################################################
# form variables handling


my $CGI=CGI->new();
my $CID=$CGI->param('cid');
my $SID=$CGI->param('sid');
my $USER=$CGI->param('user');
my $PASS=$CGI->param('pw');
my $HOST=$CGI->param('host');
my $DBNAME=$CGI->param('db');

my ($HUMAN_DATE,$TYPENUMBER);

##########################################################################################
# Main program


&head();
$HUMAN_DATE=sapiens(time());
&report();
&foot();

sub report{
                my ($i, $DBH, $STH, $NUMBEROW, $NUMBERFIELDS, @COLUMN, $REF,
                $PROTOCOL,$SERVICE, $ICMPTYPE, $ICMPCODE, $SIGID);

my $ALERT_QUERY="
SELECT event.timestamp,signature.sig_name,signature.sig_sid,sig_class.sig_class_name
FROM event,signature,sig_class
WHERE event.cid=\"$CID\" and event.sid=\"$SID\" and event.signature=signature.sig_id and
sig_class.sig_class_id=signature.sig_class_id;";

my $IPHEADER_QUERY="
SELECT inet_ntoa(iphdr.ip_src),inet_ntoa(iphdr.ip_dst),iphdr.ip_len,iphdr.ip_ttl,iphdr.ip_proto
FROM iphdr
WHERE iphdr.cid=\"$CID\" and iphdr.sid=\"$SID\" ;";

my $TCPHEADER_QUERY="
SELECT tcphdr.tcp_sport,tcphdr.tcp_dport,tcphdr.tcp_win,tcphdr.tcp_flags,tcphdr.tcp_seq
FROM tcphdr
WHERE tcphdr.cid=\"$CID\" and tcphdr.sid=\"$SID\" ;";

my $UDPHEADER_QUERY="
SELECT udphdr.udp_sport,udphdr.udp_dport
FROM udphdr
WHERE udphdr.cid=\"$CID\" and udphdr.sid=\"$SID\" ;";
```

```
my $ICMPHEADER_QUERY="
SELECT icmphdr.icmp_type,icmphdr.icmp_code,icmp_id
FROM icmphdr
WHERE icmphdr.cid=\"$CID\" and icmphdr.sid=\"$SID\" ;";

##########################################################################################
# Report

        $DBH = DBI->connect("DBI:mysql:$DBNAME:$HOST","$USER","$PASS");

        print "<H3>Alert detail for event $CID from sensor $SID :</H3>\n"; # debug
        $STH = $DBH->prepare($ALERT_QUERY); # sending query to mysql
        $STH->execute();
        @COLUMN=("DATE","SIGNATURE","SID","CLASS","INFO");

        # printing the result TAB
        print "
        <CENTER>
        <TABLE BORDER=0 CELLPADDING=5 bgcolor=\"#aeaaae\" width=\"100\%\">
        <TR bgcolor=\"#00008b\">";
        for ($i = 0;  $i < 5;  $i++) {
           print "
           <TD ALIGN=center><B><FONT size=\"1\"
color=\"#ffffff\">@COLUMN[$i]</FONT></B></TD>";
        }
        print "</TR>\n";
        while ($REF = $STH->fetchrow_arrayref) {
                print "<TR bgcolor=\"#cccccc\">";
                for ($i = 0;  $i < 4;  $i++) {
                        print "
                        <TD><CENTER><FONT
size=\"1\">$$REF[$i]</FONT></CENTER></TD>";
                }
                $SIGID=$$REF[$i-2];
                print "
                <TD ALIGN=center><A HREF=\"http://www.snort.org/snort-
db/sid.html?sid=".$SIGID."\" TARGET=new>
                <IMG BORDER=0 width=50 height=22 SRC=\"./snort.gif\"
ALT=\"Whois\"></A></TD>";

                print "</TR></TABLE></CENTER>\n";
        }

##########################################################################################
# IP layer

        $STH = $DBH->prepare($IPHEADER_QUERY); # sending query to mysql
        $STH->execute();
        $NUMBEROW = $STH->rows;
        print "<H4>IP header details</H4>\n";
        @COLUMN=("SRC IP","DST IP","LEN","IP TTL","PROTO");
```

```
         # printing the result TAB

         print "
         <CENTER>
         <TABLE BORDER=0 CELLPADDING=5 bgcolor=\"#aeaaae\" width=\"100\%\">
         <TR bgcolor=\"#00008b\" color=\"#ffffff\">";
         for ($i = 0;  $i < 5;  $i++) {
            print "
            <TD ALIGN=center><B><FONT size=\"1\"
color=\"#ffffff\">@COLUMN[$i]</FONT></B></TD>";
         }
         print "</TR>\n";
         while ($REF = $STH->fetchrow_arrayref) {
                  print "<TR bgcolor=\"#cccccc\">";
                  for ($i = 0;  $i < 4;  $i++) {
                           print "
                           <TD><CENTER><FONT
size=\"1\">$$REF[$i]</FONT></CENTER></TD>";
                  }
                  for ($i = 4;  $i < 5;  $i++) {
                           $PROTOCOL=&ipproto($$REF[$i]);
                           print "
                           <TD><CENTER><FONT
size=\"1\">$PROTOCOL</FONT></CENTER></TD>";
                  }
                  print "</TR></TABLE></CENTER>\n";
         }
##########################################################################################
# TCP layer

         if ($PROTOCOL eq "TCP"){

                  $STH = $DBH->prepare($TCPHEADER_QUERY); # sending query to mysql
                  $STH->execute();
                  print "<H4>TCP header details</H4>\n";
                  @COLUMN=("SRC PORT","DST PORT","WINDOW","FLAG","SEQUENCE");

                  # printing the result TAB

                  print "
                  <CENTER>
                  <TABLE BORDER=0 CELLPADDING=5 bgcolor=\"#aeaaae\" width=\"100\%\">
                  <TR bgcolor=\"#00008b\">";
                  for ($i = 0;  $i < 5;  $i++) {
                            print "
                            <TD ALIGN=center><B><FONT size=\"1\"
color=\"#ffffff\">@COLUMN[$i]</FONT></B></TD>";
                  }
                  print "</TR>\n";
                  while ($REF = $STH->fetchrow_arrayref) {
                           print "<TR bgcolor=\"#cccccc\">";
                           for ($i = 0;  $i < 2;  $i++) {
                                    my $PORT = $$REF[$i];
```

```
                                        if (($PORT < 1024)&&($PORT != 0)){
                                                my $SERVICE=getservbyport($PORT, 'tcp');
                                                print "
                                                <TD><CENTER><FONT
size=\"1\">$SERVICE</FONT></CENTER></TD>";
                                        }else{
                                                print "
                                                <TD><CENTER><FONT
size=\"1\">$PORT</FONT></CENTER></TD>";
                                        }

                                }
                                for ($i = 2;  $i < 5;  $i++) {
                                        print "
                                        <TD><CENTER><FONT
size=\"1\">$$REF[$i]</FONT></CENTER></TD>";
                                }
                        print "</TR></TABLE></CENTER>\n";
                        }
        }
##########################################################################################
# UDP layer

        if ($PROTOCOL eq "UDP"){

                $STH = $DBH->prepare($UDPHEADER_QUERY); # sending query to mysql
                $STH->execute();
                print "<H4>UDP header details</H4>\n";
                @COLUMN=("SRC PORT","DST PORT");

                # printing the result TAB

                print "
                <CENTER>
                <TABLE BORDER=0 CELLPADDING=5 bgcolor=\"#aeaaae\" width=\"100\%\">
                <TR bgcolor=\"#00008b\">";
                for ($i = 0;  $i < 2;  $i++) {
                        print "
                        <TD ALIGN=center><B><FONT size=\"1\"
color=\"#ffffff\">@COLUMN[$i]</FONT></B></TD>";
                }
                print "</TR>\n";
                while ($REF = $STH->fetchrow_arrayref) {
                        print "<TR bgcolor=\"#cccccc\">";
                        for ($i = 0;  $i < 2;  $i++) {
                                print "
                                <TD><CENTER><FONT
size=\"1\">$$REF[$i]</FONT></CENTER></TD>";
                        }
                print "</TR></TABLE></CENTER>\n";
                }
        }
```

```
##########################################################################################
# ICMP layer

        if ($PROTOCOL eq "ICMP"){

                $STH = $DBH->prepare($ICMPHEADER_QUERY); # sending query to mysql
                $STH->execute();
                print "<H4>ICMP header details</H4>\n";
                @COLUMN=("ICMP TYPE","ICMP CODE","ICMP ID");

                # printing the result TAB

                print "
                <CENTER>
                <TABLE BORDER=0 CELLPADDING=5 bgcolor=\"#aeaaae\" width=\"100\%\">
                <TR bgcolor=\"#00008b\">";
                for ($i = 0;  $i < 3;  $i++) {
                        print "
                        <TD ALIGN=center><B><FONT size=\"1\"
color=\"#ffffff\">@COLUMN[$i]</FONT></B></TD>";
                }
                print "</TR>\n";
                while ($REF = $STH->fetchrow_arrayref) {
                        print "<TR bgcolor=\"#cccccc\">";
                        for ($i = 0;  $i < 1;  $i++) {
                                $TYPENUMBER=$$REF[$i];
                                $ICMPTYPE=&icmptype($TYPENUMBER);
                                print "
                                <TD><CENTER><FONT
size=\"1\">$ICMPTYPE</FONT></CENTER></TD>";
                        }
                        for ($i = 1;  $i < 2;  $i++) {
                                if ($TYPENUMBER == 3){
                                        $ICMPCODE=icmp3code($$REF[$i]);
                                }
                                elsif ($TYPENUMBER == 5){
                                        $ICMPCODE=icmp5code($$REF[$i]);
                                }
                                elsif ($TYPENUMBER == 11){
                                        $ICMPCODE=icmp11code($$REF[$i]);
                                }
                                else{
                                        $ICMPCODE=$$REF[$i];
                                }
                                print "
                                <TD><CENTER><FONT
size=\"1\">$ICMPCODE</FONT></CENTER></TD>";
                        }
                        for ($i = 2;  $i < 3;  $i++) {
                                print "
                                <TD><CENTER><FONT
size=\"1\">$$REF[$i]</FONT></CENTER></TD>";
                        }
                print "</TR></TABLE></CENTER>\n";
                }
        }
        $STH->finish();
        $DBH->disconnect();
}


#####################################################################################
# DECODE FUNCTIONS

sub ipproto{

        my %table;
        my $number = $_[0];

        $table{1} = "ICMP";
        $table{2} = "IGMP";
        $table{3} = "GGP";
        $table{4} = "IP";
        $table{5} = "ST";
        $table{6} = "TCP";
        $table{7} = "UCL";
        $table{8} = "EGP";
        $table{9} = "IGP";
        $table{10} = "BBN-RCC-MON";
        $table{11} = "NVP-II";
        $table{12} = "PUP";
        $table{13} = "ARGUS";
        $table{14} = "EMCON";
        $table{15} = "XNET";
        $table{16} = "CHAOS";
        $table{17} = "UDP";
        $table{18} = "MUX";
        $table{19} = "DCN-MEAS";
        $table{20} = "HMP";
        $table{21} = "PRM";
        $table{22} = "XNS-IDP";
        $table{23} = "TRUNK-1";
        $table{24} = "TRUNK-2";
        $table{25} = "LEAF-1";
        $table{26} = "LEAF-2";
        $table{27} = "RDP";
        $table{28} = "IRTP";
        $table{29} = "ISO-TP4";
        $table{30} = "NETBLT";
        $table{31} = "MFE-NSP";
        $table{32} = "MERIT-INP";
        $table{33} = "SEP";
        $table{34} = "3PC";
        $table{35} = "IDPR";

        if ($table{$number}){
                return $table{$number};
        }else{
                return $number;
        }
}

sub icmptype{

        my %table;
        my $number = $_[0];


        $table{0} = "Echo Reply";
        $table{3} = "Destination Unreachable";
        $table{4} = "Source Quench";
        $table{5} = "Redirect";
        $table{8} = "Echo Request";
        $table{11} = "Time Exceeded";
        $table{12} = "Parameter Problem";
        $table{13} = "Timestamp";
        $table{14} = "Timestamp Reply";
        $table{15} = "Information Request";
        $table{16} = "Information Reply";

        if ($table{$number}){
                return $table{$number};
        }else{
                return $number;
        }
}

sub icmp3code{

        my %table;
        my ($number) = $_[0];

        $table{0} = "net unreachable";
        $table{1} = "host unreachable";
        $table{2} = "protocol unreachable";
        $table{3} = "port unreachable";
        $table{4} = "fragmentation needed and DF set";
        $table{5} = "source route failed";

        if ($table{$number}){
                return $table{$number};
        }else{
                return $number;
        }

}

sub icmp5code{
        my %table;
        my ($number) = $_[0];

        $table{0} = "Network";
        $table{1} = "Host";
        $table{2} = "TOS and Network";
        $table{3} = "TOS and Host";

        if ($table{$number}){
                return $table{$number};
        }else{
                return $number;
        }
}

sub icmp11code{

        my %table;
        my ($number) = $_[0];

        $table{0} = "ttl in transit";
        $table{1} = "frag reassembly";

        if ($table{$number}){
                return $table{$number};
        }else{
                return $number;
        }
}
```

Showing Purge.sh:

```
#!/bin/bash
#
# purge snort/acid database - delete alerts based on age
# Gabriel L. Somlo, 10/28/2002
#--------------------------------------------------------------------------
# Last Modified by Nitin Bhatia for Linux on 10/15/2003.
# This can be used for Linux as it is, if following my latest instructions on
# snort installation.
#
# NOTE: If You are using this script to purge on UNIX or on BSD. Please check the
# MSql Command Line path and in MySQL 4.0 and later, we should be able to do a
# multi-table delete instead
#--------------------------------------------------------------------------

# delete any events if older than ${DEL_ALL} (format is dd:hh:mm)

AGE=4:0:0
```

```
# optimize databases after purge
OPT=yes

# information required to connect to the database:
DATABASE=snort
USER=snort
PASSWORD=new_password # This is your snort database password

###### END USER CONFIGURABLE OPTIONS ######################
# Convert a string of the form "DD:HH:MM" into the corresponding nr. of seconds
to_seconds () {
 DD=${1%%:*}
 MM=${1##*:}
 HM=${1#*:}
 HH=${HM%%:*}
 let SECS=${DD}*24*60*60+${HH}*60*60+${MM}*60
 echo ${SECS}
}
EVT_AGE=$(to_seconds ${AGE})

# the mysql command line
MYSQL="/usr/local/mysql/bin/mysql ${DATABASE} -u${USER} -p${PASSWORD}"

# query for event count
CNTQUERY="select count(*) from event;\n"

# count number of events in database
TOTALCNT=$(echo -e ${CNTQUERY} | ${MYSQL} | tail +2)

# query for events to delete
DELQUERY="select sid, cid from event where unix_timestamp(now()) - unix_timestamp(timestamp) >
${EVT_AGE};\n"

# given a stream of "sid cid" pairs on stdin, delete them from the database
# and finally print out the number of deleted events on stdout
purge_database () {
 let CNT=0
 while read SID CID; do
# Changed from acid_ag_alert to acid_event
   DELETE="delete from acid_ag_alert where ag_sid=${SID} and ag_cid=${CID};\n"
   for T in iphdr tcphdr udphdr icmphdr opt data event acid_event; do
     DELETE="${DELETE}delete from ${T} where sid=${SID} and cid=${CID};\n"
   done
   echo -e ${DELETE} | ${MYSQL}
   let CNT++
 done
 echo ${CNT}
} # end purge_database

# delete eligible events from database, and count
DELCNT=$(echo -e ${DELQUERY} | ${MYSQL} | tail +2 | purge_database)
echo "DELETED ${DELCNT} of ${TOTALCNT} events"
```

```
# if optimization not requested, or we didn't delete anything, we're done here
if [ "${OPT}" != "yes" ] || (( ${DELCNT} == 0 )); then
  exit 0
fi

# Optimize all tables if requested
TABLES=""
for T in $(echo "show tables;" | ${MYSQL} | tail +2); do
  if [ "${TABLES}" != "" ]; then
    TABLES="${TABLES},"
  fi
  TABLES="${TABLES}${T}"
done
echo "optimize table ${TABLES}" | ${MYSQL}
```

REFERENCES

1. Abdullah A. Sebyala, Temitope Olukemi and Dr. Lionel Sacks, "Active Platform Security through Intrusion Detection Using Naïve Bayesian Network for Anomaly Detection", In Proceedings of the London Communications Symposium, England 2002.
2. Byung Rae Cha & Dong Seob Lee, "Network-Based Anomaly Intrusion Detection improvement by Bayesian network and Indirect relation", Lecture Notes in Computer Science, Springer Berlin/Heidelberg, vol. 4693/2009, pp. 141-148.
3. Christopher Kruegel, Darren Mutz, William Robertson, Fredrik Vaeur "Bayesian Event Classification for Intrusion Detection", IEEE Computer Society, in Proc. 19th Annual Computer Security Applications Conference, Dec 2003, pp. 14-23.
4. Daniel J. Burroughs, Linda F. Wilson and George V. Cybenko, "Analysis of Distributed Intrusion Detection System uses Bayesian", IEEE Artificial Intelligence Edition, vol. 0-7803-7371-5/2002, pp. 329-335.
5. Frederic Cuppens "Managing Alerts in a Multi-Intrusion Detection Environment", IEEE Computer Security Applications Conference 2001, ACSAC 2001 in Proc. 17th Annual SAC, 10-14 Dec 2001, pp. 22-31.
6. Huajie Zhang, Charles X. Ling and Zhiduo Zhao, "The Learnability of Naïve Bayes", Lecture Notes in Computer Science, vol. 1822, in Proc. Of 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Adv. A.I., 2000, pp. 432-441.

7. J.D. Judd, J.C. McEachen, J.B. Michael and D.W. Ettlich, "Network Stream Splitting for Intrusion Detection", IEEE Australia 11th International conference on Networks, 28 Sept. -1 Oct. 2003, pp. 525-530.

8. J. Pikoulas, W.J. Buchanan, M. Mannion, K. Triantafyllopoulos, "An Agent-based Bayesian Forecasting model for Enhancing Network Security", Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS '01), 2001, pp. 247-254.

9. Nahla ben Amor, Salem Benferhat, Zied Elouedi, "Naive Bayes vs. decision trees in intrusion detection systems", in Proc. of the 2004 ACM symposium on Applied computing, Computer Security (SEC) 2004, pp. 420-424.

10. Rafeeq U. Rehman, "Intrusion Detection with Snort", Prentice Hall Publication 2003.

11. Ricardo S. Puttini, Zakia Marrakchi and Ludov Me, "A Bayesian Classification Model for Real-Time Intrusion Detection", in Proc. AIP Conference 2003, Securite Des Systemes d'Information et Reseaux, France, 2003.

12. Snort Team, "Snort Intrusion Detection 2.1", Sygress Publication 2004.

13. Snort user's manual http://www.Snort.org/docs/writing_rules.

14. Snort FAQ http://www.Snort.org/docs/faq.html.

15. The Snort user's mailing list http://lists.sourceforge.net/lists/listinfo/snort-users.

16. Yu-Sung Wu, Bingrui Foo, Yongguo Mei, Saurabh Bagchi, "Collaborative Intrusion Detection System (CIDS)", in Proc. of 19th Annual Computer Security

Applications Conference, ACSAC, IEEE Computer Society Edition, vol. 1063-9527/2003, Las Vegas, Nevada December 08 -12 2003, pp. 234.